Smart Sequence Similarity Search (S^4) system

Zhuo Chen

SMART SEQUENCE SIMILARITY SEARCH (S^4) SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Zhuo Chen

June 2004

SMART SEQUENCE SIMILARITY SEARCH (S^4) SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Zhuo Chen

June 2004

Approved by:

[redacted]

Arturo Concepcion, Chair, Computer Science	02 Jun 2004 Date

[redacted]

Anthony Metcalf, Co-Chair, Biology

[redacted]

Klaus Brasch, Biology

[redacted]

George Georgiou, Computer Science

[redacted]

Kerstin Voigt, Computer Science

ABSTRACT

Sequence similarity searching is commonly used to help clarify the biochemical and physiological features of newly discovered genes or proteins. An efficient similarity search relies on the choice of tools and their associated subprograms and numerous parameter settings. This could be very challenging for similarity search users, especially those at the beginner level. To assist researchers in selecting optimal programs and parameter settings for efficient sequence similarity searches, we have developed a web-based expert system, Smart Sequence Similarity Search (S4). The project is implemented in Java and Jess scripts, and uses the Jess Expert System as its reasoning core. The expert knowledge provided for a sequence similarity search is represented in the form of decision tree and stored in a XML file. Our system also provides interfaces for expert users to improve this knowledge by extending the decision tree. With its capability to continuously improve sequence similarity searches through a decision tree, our web-based expert system provides a solid advising tool for researchers interested in efficient sequence similarity searches.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Arturo Concepcion, for his endless support, patience, and encouragement throughout my graduate studies. He provided a motivating, enthusiastic, and critical atmosphere during the many discussions we had. It was my great pleasure to conduct this project under his supervision. His technical and editorial advice was essential to the completion of this project. I also want to thank my co-advisor, Dr. Anthony Metcalf, for his invaluable advice and supervision on biological aspects of the project. My thanks also go to the members of my committee, Drs. Klaus Brasch, George Georgiou, Kerstin Voigt, for their astute direction, timely advice, and thoughtful comments. I also want to express my gratitude for their reading previous draft of this document and providing many valuable comments that improved the presence and contents of this document.

I would like to express my deepest gratitude to my husband Xiwei Wu for his understanding and love during the past few years. His support and encouragement was in the end that made this project possible. He also gave me great help on sequence similarity search. My parents receive my special thank and love for their dedication and support.

Finally, the support of the National Science Foundation under the award 9810708 is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PROGRAM LISTINGS

CHAPTER ONE

INTRODUCTION

1.1 Background

As we are entering the genomic era, high throughput technologies, such as large-scale sequencing, gene expression profiling, single-nucleotide polymorphism (SNP) discovery, and proteomics, become more and more developed and commonly used in laboratories. One of the net results of the advance of these technologies is the discovery of thousands of novel DNAs and proteins every month.

DNA is the basic carrier molecule of the genetic code of most organisms. An organism's total DNA complement is called its genome. DNA is usually represented by its sequence of nucleotide bases consisting of Adenosine (A), Thymine (T), Cytosine (C), and Guanosine (G). The genetic information encoded in DNA can be transcribed into mRNA, which is then translated into proteins. Protein can also be represented by a sequence of characters, which are the abbreviations of twenty different types of amino acid residues, protein's basic building block. Protein must form stable tertiary structures to be functional. Therefore, DNA and protein can both be represented by a sequence of characters.

These DNA and protein sequences are usually deposited and stored in the sequence databases and can be retrieved by the researchers. It should be noted that it is not the sequence itself, but the function of the sequence that is the most interesting. The functions of many sequences, especially from model systems such as yeast, bacteria, mouse, and human, have been determined experimentally. However, the majority of sequences in sequence databases do not have known functions, making function prediction a high demand. Consequently, sequence similarity searches against databases have become a mainstay of bioinformatics, partially because of the maturity of sequence alignment algorithms and the availability of high quality similarity search tools on the Internet.

1.1.1 Introduction to Similarity Search

Similarity searching is the application of knowledge gained from previous experiments to the problem of discovering the biochemistry and physiology of a newly discovered gene or its protein product. In practice, the sequence of interest is compared to every sequence in a sequence database, and the similar ones are identified. If a query sequence is similar to a database sequence of known function, structure, or biochemical activity, the query sequence is predicted to have the same function, structure, or biochemical activity. The strength of these predictions depends on the quality of the alignment between the sequences. Sequence alignment is to put two sequences together to find common patterns within sequences. The purpose of making alignments is to discover whether or not sequences are homologous or derived from a common ancestor gene, which implies similar function. If an alignment can be found that would rarely be observed between random sequences, the sequences are predicted to be related with a high degree of confidence.

1.1.2 Algorithms for Similarity Search

There are three major algorithms widely used in sequence similarity searching, Smith-Waterman [B1], FASTA (pronounced FAST-Aye) [B2], and BLAST (Basic Local Alignment Search Tools) [B3]. The different algorithms add different restrictions to the simple model of sequence evolution on which similarity searching is based.

Smith-Waterman is the most rigorous algorithm and does not place any heuristic restrictions on the evolutionary model [B1]. It is mathematically rigorous, and guaranteed to find the best scoring alignment between the pair of sequences being compared [B1]. FASTA stands for FAST-ALL, reflecting the fact that it can be used for a fast protein comparison or a fast nucleotide comparison. The high speed of this program is achieved by using the observed pattern of word hits to identify potential matches before attempting the more time consuming optimized search. Not every word hit is investigated, instead the program initially looks for segments containing several nearby hits [B4]. The BLAST programs are a set of sequence comparison algorithms used to search sequence databases for optimal local alignments to a query. The BLAST algorithm uses a word based heuristic similar to that of FASTA [B3]. The BLAST programs improved the overall speed of searches while retaining good sensitivity by breaking the query and database sequences into fragments ("words"), and initially seeking matches between fragments. Word hits are then extended in either direction in an attempt to generate an alignment [B5].

Both BLAST and FASTA place additional restrictions on the alignments that they report in order to speed up their operation. The actual pattern of evolutionary changes between the query sequence and any homologues in the database can be incompatible with the heuristic restrictions imposed by either BLAST or FASTA. Alternatively the additional selectivity that results from these restrictions can sometimes be an advantage. Because of the mathematical rigor and lack of restrictions, the Smith-Waterman algorithm is more sensitive than either BLAST or FASTA [B6]. This additional sensitivity comes at the price of being a very much slower way to search a sequence database than are either BLAST or FASTA [B6].

1.1.3 Online Tools for Similarity Search

BLAST programs are available interactively through a large server at the National Center for Biotechnology Information (NCBI) (http://www.ncbi.nlm.nih.gov). FASTA has recently become available interactively at the European Bioinformatics Institute (EBI) (http://www.ebi.ac.uk/fasta33/). Results can also be sent to users by email from FASTA server. Because the Smith-Waterman algorithm is slow, there are very limited implementations of this algorithm for large scale similarity searches. However, there are efforts to better implement the Smith-Waterman algorithm [B7]. Right now, several servers provide similarity search services based on Smith-Waterman algorithm by emailing the results to the user, but interactive online tools is still not available. The largest Smith-Waterman based similarity search server with the most functionality is provided by DNA Data Bank of Japan (DDBJ). The program is called S&W SEARCH (http://www.ddbj.nig.ac.jp/E-mail/homology.html).

1.1.4 Parameter Settings in Similarity Search

All of the above online tools provide a set of programs for different searches based on the type of query sequences and database sequences. For example, BLASTP is for searching protein databases with a protein query sequence, while TBLASTN is for searching translated DNA databases with a protein query. Some of the tools also provide programs designed to be used in special cases. For example, FASTS3 is to compare linked peptides from mass-spectrometry of a protein to a protein database, while FASTF3 is to compare mixed peptides obtained by Edman degradation of a CNBr cleavage of a protein to a protein database.

There are also several parameters that need to be set before initiating a similarity search. Among these, scoring matrices and gap penalty are probably the most important parameters [B8] [B9]. While a unitary matrix is usually used for DNA pairs, amino acid substitution matrices are used for protein alignments. These substitution matrices are matrices in which each possible residue substitution is given a score reflecting the probability that it is related to the corresponding residue in the query [B10]. The alignment score will be the sum of the scores for each position. Various scoring systems (e.g. PAM, BLOSUM and PSSM) for quantifying the relationships between residues can be used. Depending on the purpose of the search, different substitution matrices should be chosen to optimize the results [B8]. For example, BLOSUM62 is commonly used if a protein sequence of 100 amino acids matched against a protein database to reveal any similar sequences. PAM matrices are based on a specific revolution model and commonly used when closely related search hits are expected [B10]. Word size in BLAST and ktup in FASTA can greatly influence the search speed and sensitivity [B11;B3]. Increasing the word size or ktup decreases the number of potential hits to search. This results in greater speed, but with the tradeoff of lower sensitivity. The choice of databases to search against is another commonly used parameter. For example, comprehensive databases, such as nr in NCBI and uniprot in EBI, should be used if the user wants to get as many similar sequences as possible. Specialized databases that are frequently updated might be particularly useful if the user routinely searches them with the same query every month. Some users also might want to restrict the search within one or several types of species, EST database, raw sequence database, or genome databases depending on their interests.

There are also tool-specific parameters. For example, FASTA allows the users to restrict the length of database sequences to search with, which is particularly useful when the user is only interested in looking for members of a specific protein family. Similarly, BLAST allows users to perform the search with user-defined position specific scoring matrix (PSSM). The choice of which tools to use can also depend on the sensitivity or selectivity the user desires, because of their different performance in Section 1.1.2.

1.1.5 Expert Systems

An expert system is a program, which attempts to mimic human expertise by applying inference methods to a specific body of knowledge called the domain [B12]. It is a branch of Artificial Intelligence (AI). AI can be defined as the field of study which is attempting to build systems which if attempted by people would be considered intelligent. AI is a broad field with many application areas, including but not limited to natural language, robotics, speech, understanding, vision, artificial neural systems, and expert systems [B12]. Among them, expert system provides a very successful solution to the classic AI problem.

Expert systems have been applied to virtually every field of knowledge in research, business and industry. These fields include chemistry, electronics, medicine, engineering, geology, computer, etc. As an example, MYCIN is the most well known medical expert system [B13]. It was developed at Stanford in the 1970s and aim to diagnose and recommend treatment for certain blood infections. The success of MYCIN demonstrated that AI could be used for practical real-world problem. Another notable example in medicine is PUFF, which diagnoses the presence and severity of lung disease in patient by interpreting measurements from respiratory tests administered in a pulmonary function laboratory [B14]. One example in routine business use is the XCON system developed by Digital Equipment Corporation (DEC) [B15]. It is an expert configuration system for DEC computer systems. In engineering, DELTA was developed by the General Electric Company to help railroad personnel diagnose maintenance problems and prescribe appropriate maintenance action for GE's diesel-electric locomotives [B16]. In chemistry, DENDRAL can enumerate every possible organic structure that satisfies the constraints apparent in the data by systematically generating partial molecular structures consistent with the data and then elaborating them in all possible ways [B17].

The expert system is suitable for domains that are well-bounded, but the solutions are uncertain and must be based on experience. The domain of our project is sequence similarity search, which is a well-bounded problem area. The program to use and parameter settings for the optimal search result are mainly based on experiential knowledge. These settings also might change as more knowledge accumulates. Therefore, sequence similarity searches can be considered as a potential domain for expert system.

1.2 Purpose of the Project

The choice of tools and parameters could become very challenging for similarity search users, especially for those who are at the beginner level. Actually, even experienced users can't take full advantages of these tools, because they are usually not familiar with all the parameters available to them, nor exactly how the settings would influence the results.

We intend to provide a user-friendly tool for users who are willing to perform sequence similarity searches, but feel uncomfortable with the parameter settings and not confident of which programs and tools to use. Ruled-based expert system was utilized to provide expert knowledge to help users choose the best programs with the most reasonable settings, so that optimal search results can be obtained based on the users-provided search purposes.

1.3 Organization of this Documentation

The remaining sections of this documentation will be organized as follows: Chapter 2 describes the software requirements specification. Chapter 3 provides a description of the system architecture and detailed design. Chapter 4 describes populating the decision tree. Chapter 5 is the system test. Chapter 6 is the maintenance and users manual. Finally, Chapter 7 concludes the project and lists suggestions for future developments.

CHAPTER TWO

SOFTWARE REQUIREMENTS SPECIFICATION

2.1. Introduction

2.1.1 Purpose

This Software Requirements Specification (SRS) documents the agreements concerning the purpose, characteristics and specific requirements of the proposed Web-based CSUSB sequence similarity search expert system - S^4 (Smart Sequence Similarity Search) System. The system will be developed by Zhuo Chen as the requirements for Master Degree in Computer Science of the Department of Computer Science, California State University, San Bernardino, CA.

2.1.2 Scope

Genomics and proteomics are two of the most rapidly advancing areas of molecular biology. Loosely defined, genomics is concerned with sequencing and analyzing the total genetic make up or genome of organisms, and so deals with the structure and composition of DNA and RNA. Proteomics, on the other hand, is concerned with the full characterization of the actual products of genes, namely protein. Both areas are increasingly turning to computational biology both to handle the enormous quantities of data involved and to facilitate comparison of structural and evolutionary relationship among these molecules.

The genome of any organism consists of its entire DNA complement. DNA molecules consist of two chains of only four types of nucleotides and are usually represented as a sequence of characters (A, T, C, or G). DNA encodes genetic information in the form of multiple nucleotides which can be transcribed into mRNA which is then translated into proteins. Proteins are polymers composed with at least twenty different types of amino acids. These basic subunits of proteins are linked by peptide bonds into chains of amino acids which then form tertiary structures based on their properties and interactions between them. Proteins can also be represented as a sequence of characters, with each character representing one type of amino acid.

The sequence itself is not informative; it must be analyzed by comparative methods against existing databases to develop hypothesis concerning relatives and function. Analyzing the similarities and differences of two sequences (called as pairwise alignment), at the level of individual bases or amino acids, can infer structural, functional, and evolutionary relationships among the sequences under study. With the number of sequences available for comparison growing explosively, comparison of one sequence to the entire database of known sequences becomes an important discovery technique useful to all molecular biologists. It is the shortest and surest path to study a newly discovered gene. The basic operation is to sequentially align a query sequence to each subject sequence in the database. The results are reported as a ranked hit list followed by a series of individual sequence alignments, plus various scores and statistics, which are used to decide whether the alignment is more likely to have occurred because the sequences are related or just by chance[6].

At present, the most widely used sequence similarity search tools include Smith-Waterman (SW) algorithm [B1] based program and two heuristic algorithms based programs, BLAST [B3] and FASTA [B2]. Considering sorts of algorithm, scoring system, statistical methods and searching speed and accuracy, it is hard to decide which of the tools mentioned above is the best, since each one has distinct features. Moreover, the chosen sequence database and various optional parameters, such as expect (E-value), filter (low-complexity), substitution matrix, gap cost, in each program can also have strong impact on the effectiveness of a search and usually result in highly variable results. It is always complicated and difficult for biologists, especially for those who do not have strong statistical and computer background, to set up advanced option in programs. Thus, a significant amount of time has to be invested before users can correctly choose programs and parameters for different searching purposes.

This document specifies the software requirements for functions of the smart sequence similarity search (S^4) to be used at CSUSB. It is one of the projects related to bioinformatics being developed at CSUSB in order to provide a user friendly system that facilitates DNA and protein similarity searches for students and researchers who are not experts. It will be implemented to:

- Allow users to input a sequence, either DNA or protein;
- Provide users with a series of questions with related options to select;
- Based on the user selections, suggest appropriate search program and parameters for their particular needs;

- Allow users to repeat step 2 to 3 if other options are selected that are based on different requirement on the same sequence search as input in step 1;
- Provide interfaces for users to extend the decision tree if the suggestions given in step 3 is not desired and users want to improve the services for later users;
- Provide users with remote service page for the suggested program on a separate window;
- Provide help pages to assist users for more efficient usage of the service.

The overall goal of this project is to assist bioinformatics researcher in selecting program and optimal parameters for their specific interests, needs and search requirements. In order to eliminate the expensive or specialized hardware requirements for storing the huge databases and running searching programs in our system, we only provide a list of suggested parameter settings and the option for users to open a new window connected with the remote servers. The trade-off is that our system has to depend on other servers. Thus, system maintenance is necessary to update our system based on the remote server. Moreover, because of the limitation to the knowledge of properly parameters chosen in the programs, the decision made by the system might not be the optimal one, especially at the initial stage, which requires continuous improvement on the expert system knowledge based on which the advises are made. Therefore, we also provide the functionality that allows the users to extend the system to update the knowledge of optimal parameter settings.

2.1.3 Definitions, Acronyms, and Abbreviations

- Amino Acid

  Amino acids are structural molecular of proteins. Each molecule contains both an amino group and a carboxyl group. Those that serve as the building blocks of proteins are alpha amino acids, having both the amino and carboxyl groups to the same carbon atom. 20 of these amino acids are common in proteins.

- Amino Acid Sequence

  The sequence or order of linkage of amino acids in a given polypeptide chain or protein. This is ultimately determined by the genetic code.

- Bioinformatics

  A rapidly developing branch of biology and uses techniques and concepts from informatics, statistics, mathematics, chemistry, biochemistry, physics, and linguistics. It derives knowledge from computer analysis of biological data. These can consist of the information stored in the genetic code, but also experimental results from various sources, patient statistics, and scientific literature. Research in bioinformatics includes method development for storage, retrieval, and analysis of the data.

- BLAST - Basic Local Alignment Search Tool

  It is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or DNA.
- CSS - Cascade Style Sheet

  A style sheet format for HTML documents endorsed by the World Wide Web Consortium.
- CSUSB - California State University, San Bernardino
- DNA - Deoxyribonucleic acid

  A polymer of covalently linked deoxyribonucleotides serving as the primary genetic material of most biological organisms. DNA is usually a double stranded helix of two polynucleotide chains linked by hydrogen bonds.
- DECISION TREE

  Decision trees possess (typically) a single root, a set of branches from that root, interior nodes that also have branches and terminal nodes which present the classification information. Each node within the tree represents a decision point that determines which subsequent branch is followed.

- EXPERT SYSTEM

  An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant expertise for their solutions.

- FASTA

  FASTA (pronounced FAST-Aye) stands for FAST-All, is to be used for a fast protein or nucleotide comparison.

- Gene

  Region of DNA that controls a discrete hereditary characteristic, usually corresponding to a single polypeptide, protein or RNA. It includes regions preceding and following the coding region as well as intervening sequences (intron) between individual coding segments (exon).

- Genome

  The total gene complement, comprising the genetic information of the entire organism.

- HTML - Hyper Text Markup Language

- HTTP - Hyper Text Transfer Protocol

  The client/server protocol that defines how messages are formatted and transmitted on the World Wide Web
- IEEE - Institute of Electrical and Electronics Engineers
- Java

  An object oriented language developed by Sun Microsystems. Java programs are capable of running on most popular computer platforms without the need for recompilation.
- JSP - Java Server Page

  An extension to the Java servlet technology from Sun that provides a simple programming vehicle for displaying dynamic content on a Web page.
- Java Servlet

  A Java application that runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing.
- JavaScript

  A scripting language that is widely supported in Web browsers and other Web tools. It adds interactive functions to HTML pages, which are otherwise static.

JDBC (Java DataBase Connectivity) A programming interface that lets Java applications access a database via the SQL language.

- JESS

  An expert system shell and scripting language written entirely in Sun Microsystem's Java language. It supports the development of rule-based expert systems which can be tightly coupled to code written in the powerful, portable Java language.

- JRE - Java Runtime Environment
- JDK - Java Development Kit
- Nucleotide

  The structural components of nucleic acids, including DNA and RNA. They consist of a pentose sugar, a phosphate and a nitrogenous base - Adenine, Cytosine, Guanine, Thymine (for DNA) or Uracil (for RNA).

- Nucleotide Sequence

  The sequence or order of linkage of nucleotides in a given polynucleotide chain or nucleic acid. They may include both coding and non-coding sequences in DNA and RNA.

- OS - Operating System
- Protein

A complex molecule consisting of a particular sequence of amino acids (peptides) that are joined to form a protein (polypeptides). They vary in structure according to their function.

- Proteomics

  The study and analysis of protein structure and function. Becoming quite an important science with the mapping of several genomes, including the human one, and the discovery of new proteins.

- RNA - Ribonucleic acid

  It is a polymer formed from covalently linked ribonucleotide monomers.

- Rule-based Expert System

  An expert system within which the knowledge base contains the domain knowledge needed to solve problems coded in the form of rules.

- Sequence Alignment

  The procedure of comparing two or more protein and nucleic acid sequences by looking for a series of individual characters or character patterns that are in the same order in the sequences.

- SRS - Software Requirements Specification
- S^4 - Smart Sequence Similarity Search

- Smith-Waterman algorithm

  It uses dynamic programming to find optimal local alignments between sequences.
- URL - Universal Resource Locator
- XHTML - Extensible HTML

  The combining of HTML 4.0 and XML 1.0 into a single format for the Web. XHTML enables HTML to be extended with proprietary tags.

2.1.4 Overview

Section 2.2 follows the guidelines of IEEE Std. 830-1998 IEEE Recommended Practice of Software Requirements Specifications [B18]. This section provides product perspective, a summary of product functions, a description of the characteristics of the expected users, and a list of assumptions and dependencies.

Section 2.3 presents the specific requirements for this system. They are organized by mode, following the SRS Section 3 template shown in IEEE Std 830-1998, Annex A, Paragraph A.3 [B18].

2.2. Overall Description

Currently, the widely used programs on sequence similarity search are BLAST, FASTA and Smith-Waterman (SW) algorithm based programs.

SW algorithm is a local alignment algorithm developed by T.F. Smith and M.S. Waterman [B1]. It uses dynamic programming to determine how an optimal alignment between the query sequence and a database sequence can be produced. This alignment is obtained by determining what transformations the query sequence would need to undergo to match the database sequence. Transformations include substituting one character for another and inserting or deleting a string of characters. A score is assigned for each character-to-character comparison--positive scores for exact matches and some substitutions, negative scores for other substitutions and insertions/deletions. The first character in an insertion or deletion gap is scored with a gap open penalty and subsequent characters are scored with a gap extension penalty. Scores are obtained from statistically-derived scoring matrices. The combination of transformations that results in the highest score is used to generate an alignment between the query sequence and database sequence.

BLAST (Basic Local Alignment Search Tool) is a heuristic algorithm first released in 1990 by the National Center for Biotechnology Information (NCBI) [B5;B3]. The original BLAST compares very short segments of the query and database sequences in search of alignments that exceed a particular score. No insertions or deletions are considered during this process. If the required score is exceeded, BLAST investigates whether the aligned segments can be lengthened to produce a higher-scoring alignment. Now, new and improved versions of BLAST (BLAST2) have been released [B5]. These newer releases consider insertions and deletions to some extent producing better results. It offers several search types: DNA to DNA, protein to protein, and translation searches.

FASTA is a family of heuristic algorithms developed by William Pearson of the University of Virginia [B2]. It lies between BLAST and Smith-Waterman in terms of both accuracy and speed. An optimized FASTA option makes use of Smith-Waterman for part of the alignment process [B19;B20]. The FASTA family includes DNA to DNA, protein to protein, and translation searches.

2.2.1 Product Perspective

Our S^4 will be Web-based system using JESS expert system shell as logic processing core, and will be built on top of the existing well-established tools. The user interfaces will be via Internet.

The hardware interface requirement is that it must run on the existing web server. The software interface is that it must support current versions of Netscape &

Internet Explorer. The communications interface requires support for Hyper Text Transfer Protocol (HTTP).

The system will be operated 24 hours per day, 7 days per week. All actions are user- initiated. No separate backup and recovery or maintenance functions are required as that is handled by system administration on the hosting server machine.

2.2.1.1 System Interfaces. The system is a 4-tier distributed architecture (Figure 2.1):

1. The first tier is the presentation tier that displays the user interface in a web browser via HTML.

2. The second tier is the web server that uses Java Servlet or JSP pages, which can automatically created Servlet, to handle requests from the client and response to the client after logic processing by the system application. The HTTP server is provided by Apache Tomcat, which also implements JSP1.4 and Servlet 2.3 API. JESS expert system, the knowledge-based system, will be mainly responsible for the logic processing.

3. Data tier - XML document will be also provided on the server side as knowledge storage method, if required, to store expert knowledge on sequence similarity search.

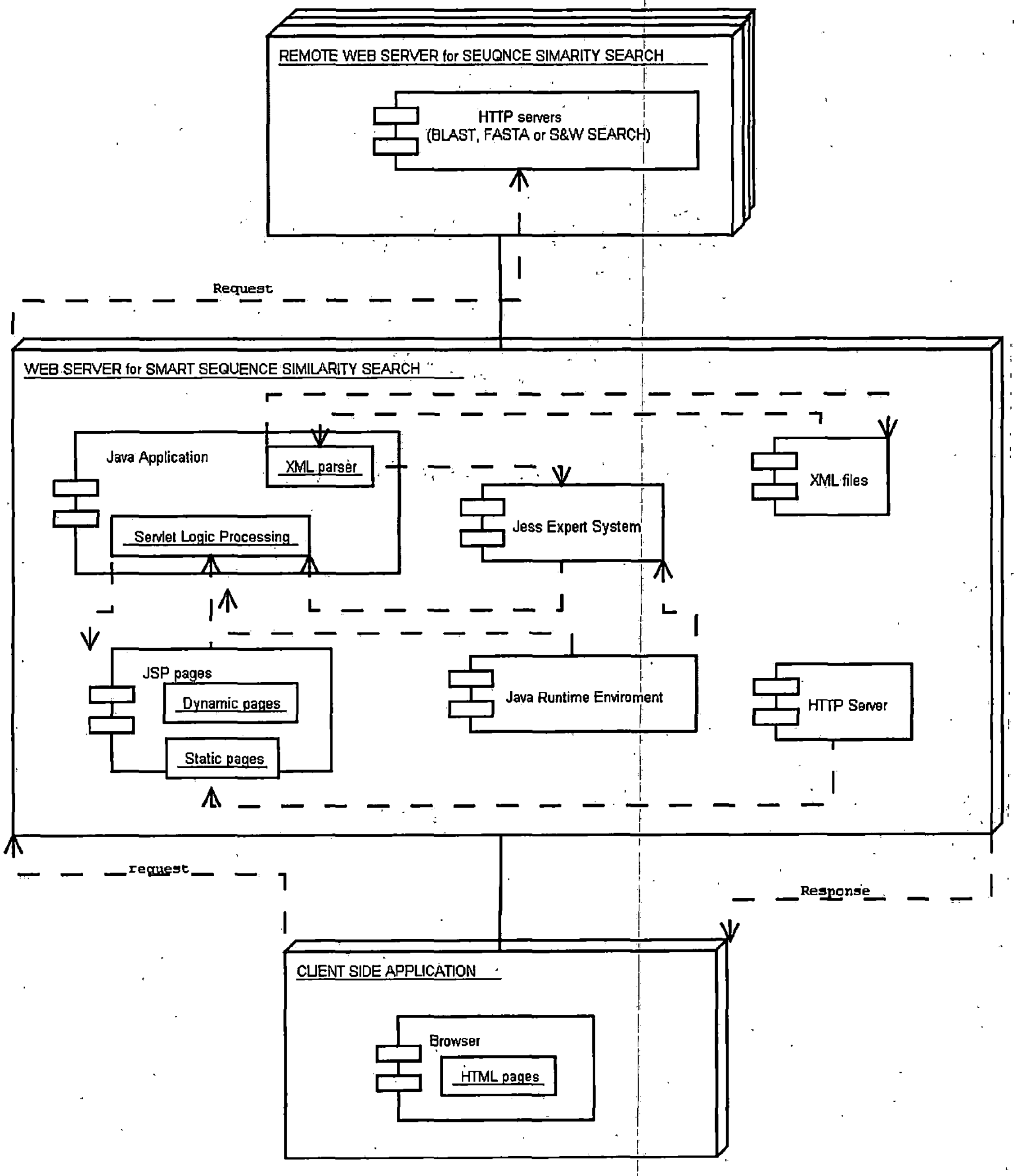


Figure 2.1. S^4 Deployment Diagram

4. Upon JESS determining the final program and settings, system web server will communicate with the forth tier - one of the remote sequence similarity

servers, including BLAST, FASTA, and S&W SEARCH. A new thread will be opened with the remote search form filled in with the determined settings, if the remote service supports. The remote servers will be one of the well-established sequence similarity search servers available online.

A user task will be conducted in the following steps:

- User initiates a user interface designed by HTML through web browser;
- Request is transmitted to web server via HTTP protocol;
- Web server responds to the request and execute a Servlet or a Servlet compiled from a JSP page;
- Servlet spawns and communicates with a separate thread running JESS expert system on the server;
- Jess expert system retrieving knowledge from XML, if necessary, and determine the information to be used by Servlet by proper reasoning.
- Servlet generates custom HTML documents containing information obtained from Jess and send it back to the user via HTTP protocol;
- HTML page is displayed by user's web browser.

2.2.1.2 User Interfaces. User interfaces for the sequence similarity search tool will be designed in HTML page. The contents are generated dynamically by Servlet or JSP in response to the user's requests. The following features will be incorporated to produce a more descriptive representation of the interface:

2.2.1.2.1 Introduction Page. This page gives a brief introduction of the system.

2.2.1.2.2 Sequence input page. Page allows users to input a DNA or protein sequence.

2.2.1.2.3 User Option Selection Pages. A number of pages allow the users to choose answers from a list of options. Each page has different options based on the previous choices the user made.

2.2.1.2.4 Result Page. Page displayed the optimal program and parameter settings the expert system chooses.

2.2.1.2.5 System Knowledge Improvement Pages. Pages allow the user to improve current knowledge in the system.

2.2.1.2.6 Help Pages. Pages provide basic knowledge for the key words used in the sequence similarity search to help user with a better concept understanding.

2.2.1.3 Hardware Interfaces. All hardware interfaces will be provided by the operating system. This system will not implement any hardware interface.

2.2.1.4 Software Interfaces. Software interfaces are provided in Java 1.4 APIs or higher, Servlet API, JSP APIs and JESS 6.1 APIs or higher. In addition, user needs a Java Script compatible web browser, (Netscape 4.0 or higher, Internet Explorer 4.0 or higher). Web server needs to implement Java Servlet API and JSP API, XML API, and provides HTTP service.

2.2.1.5 Communications Interfaces. The communication interface uses HTTP for general information. Communication between threads uses Java piped input and output stream.

2.2.1.6 Memory Constraints. There is no specific memory requirement for client computer. Although there is no explicit memory requirement for Web server, enough memory is required to guarantee acceptable service speed because of the large size memory usage of Jess expert system. 128 Mega bytes or higher are recommended.

2.2.1.7 Operations. User shall access the system through the World Wide Web. The session can be expired if the user is inactive for a period of time, and subsequently accessing any pages might be redirected to the initial page.

2.2.2 Product Functions

Figure 2.2 shows Use Case diagram that graphically depicts the users and principal functions of this system.

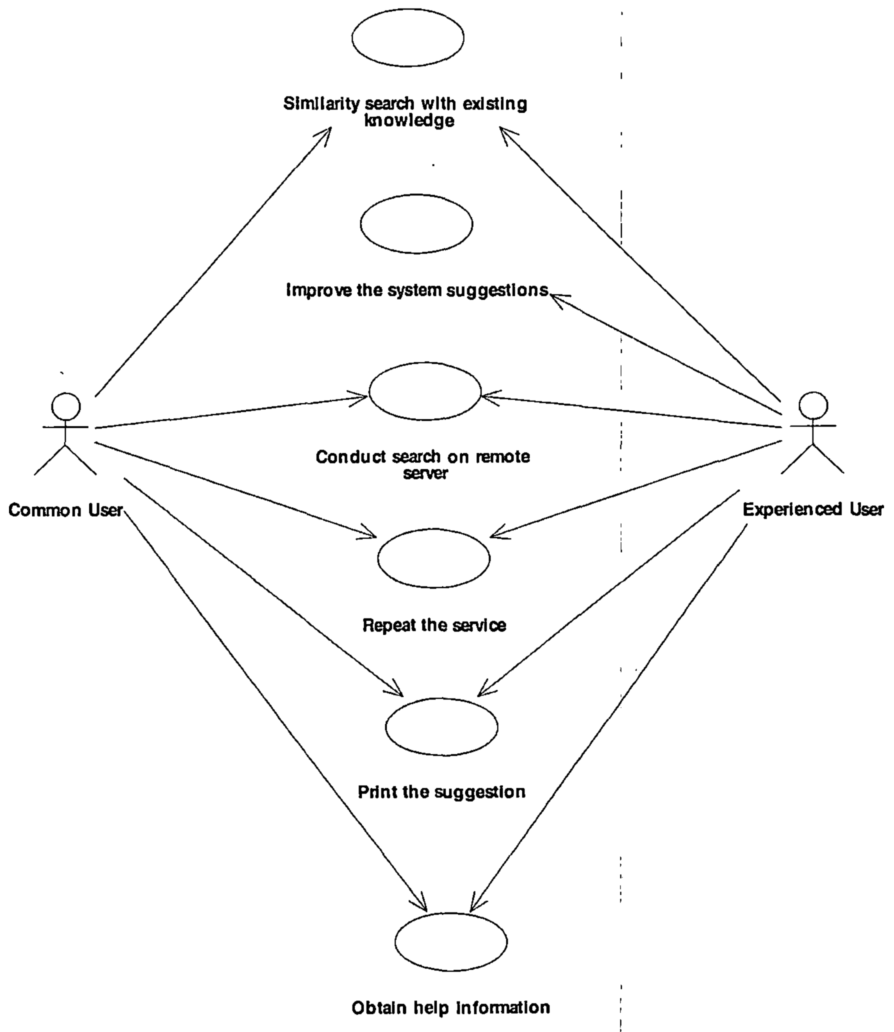

Similarity search with existing knowledge
Improve the system suggestions
Conduct search on remote server
Common User
Experienced User
Repeat the service
Print the suggestion
Obtain help information

Figure 2.2. S^4 Use Case Diagram

The functions are further described in the following subsections 2.2.1 to 2.2.5, and the actors in the diagram are further described in section 2.3.

2.2.2.1 Sequence Similarity Search Using Existing Knowledge. Users will be asked to input a sequence initially, which is followed by a series of pages. Each page contains a question and several related options. Users must choose one of the options which will determine the content to be displayed on the next page. Finally, the suggested program and associated parameter settings will be returned to the users via HTML file. In this page, several other functions will also be provided to the users and listed in 2.2.2.2-2.2.2.6.

2.2.2.2 Sequence Similarity Search with the Suggested Settings. Users have the option to use the suggested program and parameter settings to conduct sequence similarity search on the remote server. A new window will be opened and connected to the remote sequence similarity search server and allows the user to conduct the search. Whether the suggested parameter setting can be set on the remote server depends on the remote server's supports.

2.2.2.3 Repeat the Service with the Same Sequence. Users can repeat going through the similarity search option pages if they want to try other options with the same sequence input. This is for users who want to do the same sequence search with different search requirement.

2.2.2.4 Improve the System Knowledge. Experienced users can improve the service when the current suggestion provided is not satisfactory. This requires the user to input a series of questions and its related options. Each new option must be followed by another set of question and options until a final more accurate suggestion is given.

2.2.2.5 Print the Result. All users can print the result page to save the parameter settings and sequence input information for future reference.

2.2.2.6 Obtain Help Information. Users can get basic supporting information on each page by clicking a hyperlink provided in each page. These information aims to help users to understand basic concept and assist users to choose their desired options.

2.2.3 User Characteristics

Users in the system are biological researcher who would like to do the sequence similarity search, but are not familiar with the existing programs and their parameter settings. The experts on similarity search can also use the service to input information to the system for better services. All the users are assumed to know how to use web browser and speak English. They should also be able to follow the manual written in plain English.

2.2.4 Assumptions and Dependencies

These requirements assume there are no applicable hardware limitations. It is also assumed that system administration and maintenance issues will be dealt with by the host system.

2.3 Specific Requirements

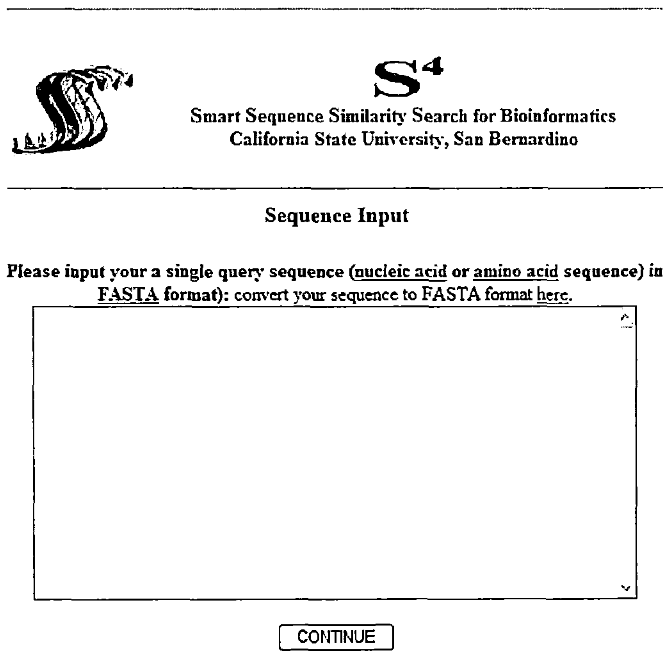
This section contains the software requirements to a level of sophistication that would enable designers to design the S^4 in conformance with the requirements of this Specification Requirement Specification document. This level of sophistication will also enable testers to generate tests for the system, to verify whether it meets the requirements. Every stated requirement will be externally perceivable by users through the usage of sample screen dumps. The requirements include a description of every input, every output, and all functions performed by the system to generate output in response to input.

2.3.1 External Interface Requirements

This is a detailed description of all inputs and outputs of S^4 system.

2.3.1.1 Introduction Page. This page is the start page for the system that gives a brief introduction to the sequence similarity search and the function of the system.

2.3.1.2 Query Sequence Input Page. This page allows the researcher to input his/her query sequence, either protein sequence or DNA sequence in FASTA format (Figure 2.3).


S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

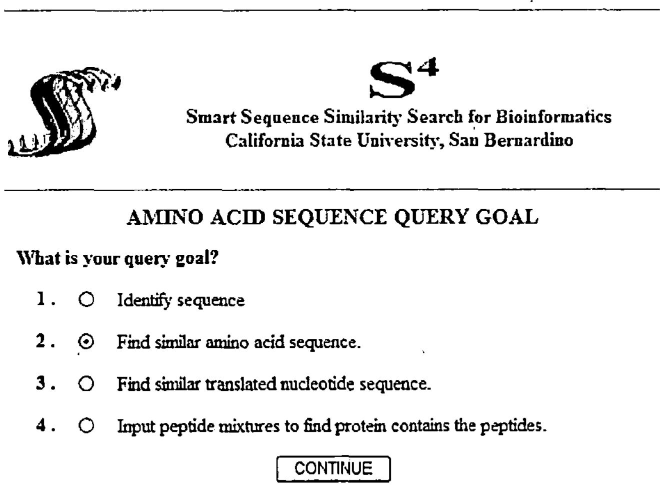
Sequence Input

Please input your a single query sequence (nucleic acid or amino acid sequence) in FASTA format): convert your sequence to FASTA format here.

Figure 2.3. Query Sequence Input Page

2.3.1.3 Pages for Users to Select Search Options. These pages are a series of selection pages that allow the researchers to select one search option each page, and direct them to the next selection page based on what they choose in the previous page (Figure 2.4 - 2.6). The pages continue to appear until the suggested parameter settings can be determined by the system. These pages are dynamically generated based on the decision tree stored in XML file and the user's selections.

2.3.1.4 Page for Displaying the Suggested Settings. This page provides the researcher the name of the suggested program and its associated parameter settings (Figure 2.7). It also contains a link to the remote sequence similarity search server if the server supports parameter settings within an URL address.

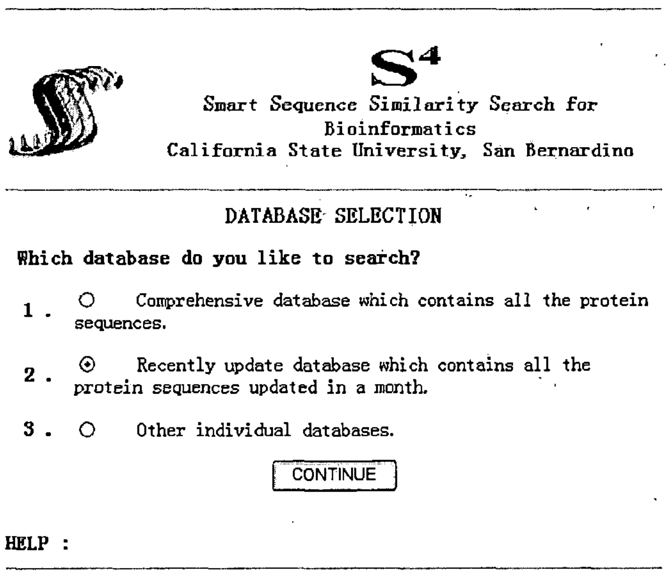

S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

AMINO ACID SEQUENCE QUERY GOAL

What is your query goal?

1. ○ Identify sequence
2. [✓ Find similar amino acid sequence.]
3. ○ Find similar translated nucleotide sequence.
4. ○ Input peptide mixtures to find protein contains the peptides.

Figure 2.4. Query Goal Page


S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

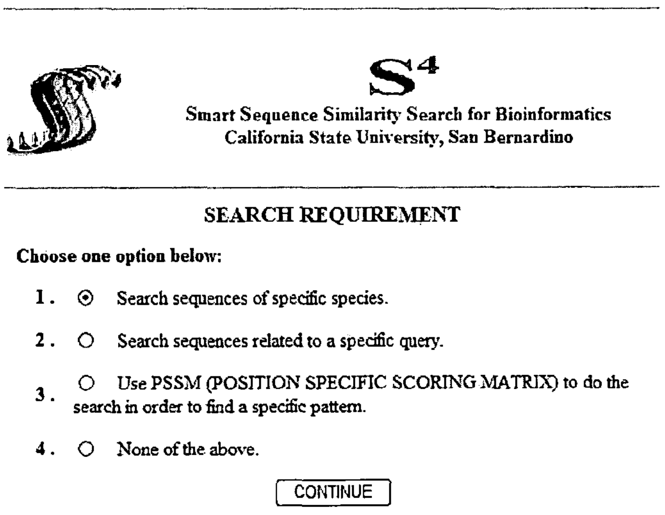
DATABASE SELECTION

Which database do you like to search?

1. ○ Comprehensive database which contains all the protein sequences.
2. [✓ ⊙] Recently update database which contains all the protein sequences updated in a month.
3. ○ Other individual databases.

HELP :

Figure 2.5. Database Selection Page

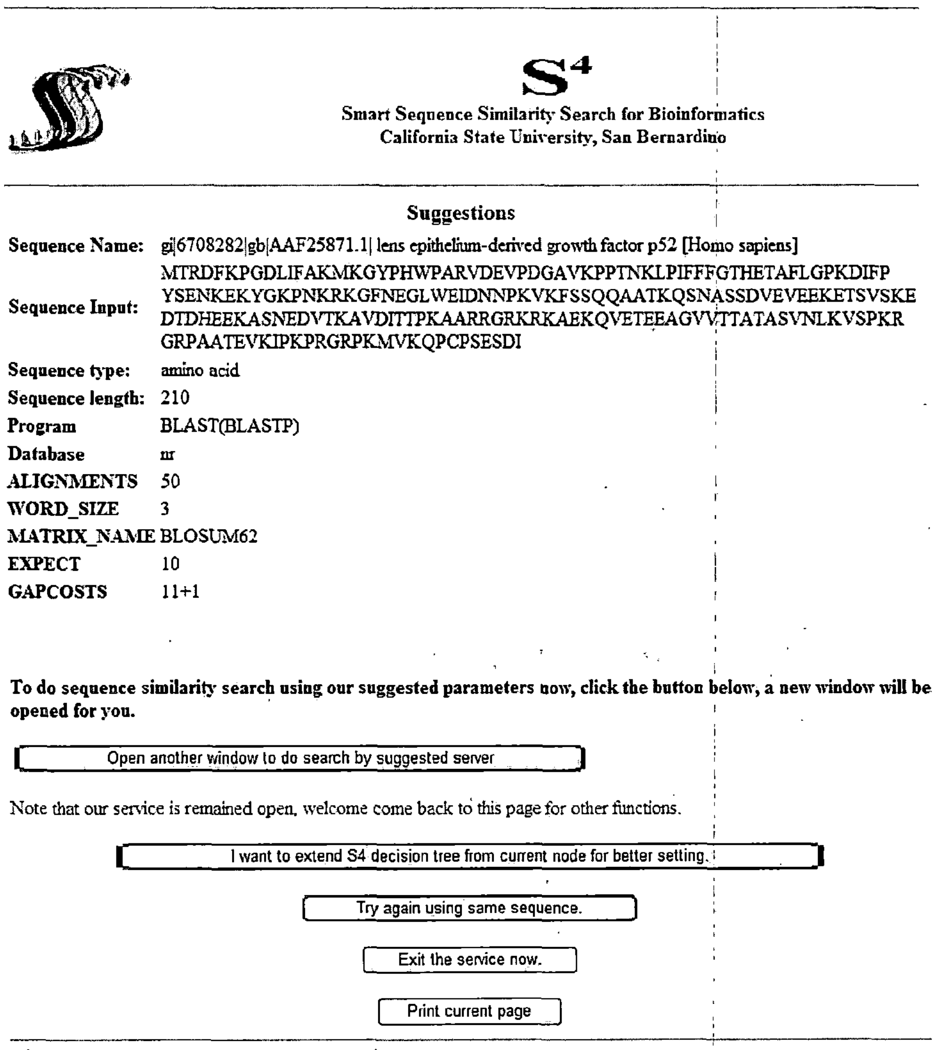

S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

SEARCH REQUIREMENT

Choose one option below:

1. [✓ ⊙] Search sequences of specific species.
2. ○ Search sequences related to a specific query.
3. ○ Use PSSM (POSITION SPECIFIC SCORING MATRIX) to do the search in order to find a specific pattern.
4. ○ None of the above.

Figure 2.6. Search Requirement Page



S4

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Suggestions

Sequence Name:	gi|6708282|gb|AAF25871.1| lens epithelium-derived growth factor p52 [Homo sapiens]
Sequence Input:	MTRDFKPGDLIFAKMKGYPHWPARVDEVPDGAVKPPTNKLPIFFFGTHETAFLGPKDIFP YSENKEKYGKPNKRKGFNEGLWEIDNNPKVKFSSQQAATKQSNASSDVEVEEKETSVSKE DTDHEEKASNEDVTKAVDITTPKAARRGRKRKAEKQVETEEAGVVTTATASVNLKVSPKR GRPAATEVKIPKPRGRPKMVKQPCPSESDI
Sequence type:	amino acid
Sequence length:	210
Program	BLAST(BLASTP)
Database	nr
ALIGNMENTS	50
WORD_SIZE	3
MATRIX_NAME	BLOSUM62
EXPECT	10
GAPCOSTS	11+1

To do sequence similarity search using our suggested parameters now, click the button below, a new window will be opened for you.

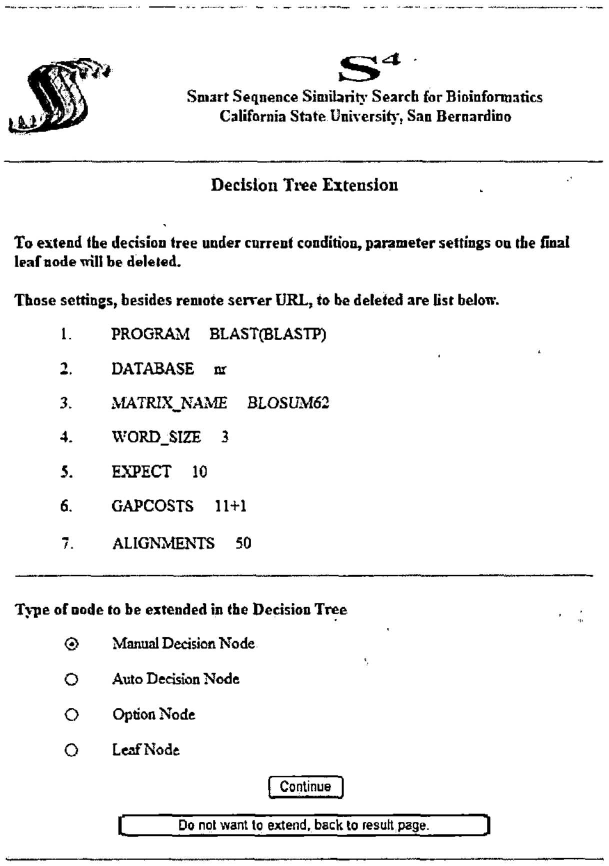
Note that our service is remained open, welcome come back to this page for other functions.

I want to extend S4 decision tree from current node for better setting.
Try again using same sequence.
Exit the service now.
Print current page

Figure 2.7. Suggestion Page

<u>2.3.1.5 Pages for Improving the System Knowledge</u>.
These pages allow the users to improve the system knowledge when they believe that the provided suggestion is not satisfactory (Figure 2.8). These pages are particularly designed for experienced researchers who grasp advanced knowledge for sequence similarity search.

These pages continue to appear until the new optimal suggestions are reached.


S4

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Decision Tree Extension

To extend the decision tree under current condition, parameter settings on the final leaf node will be deleted.

Those settings, besides remote server URL, to be deleted are list below.

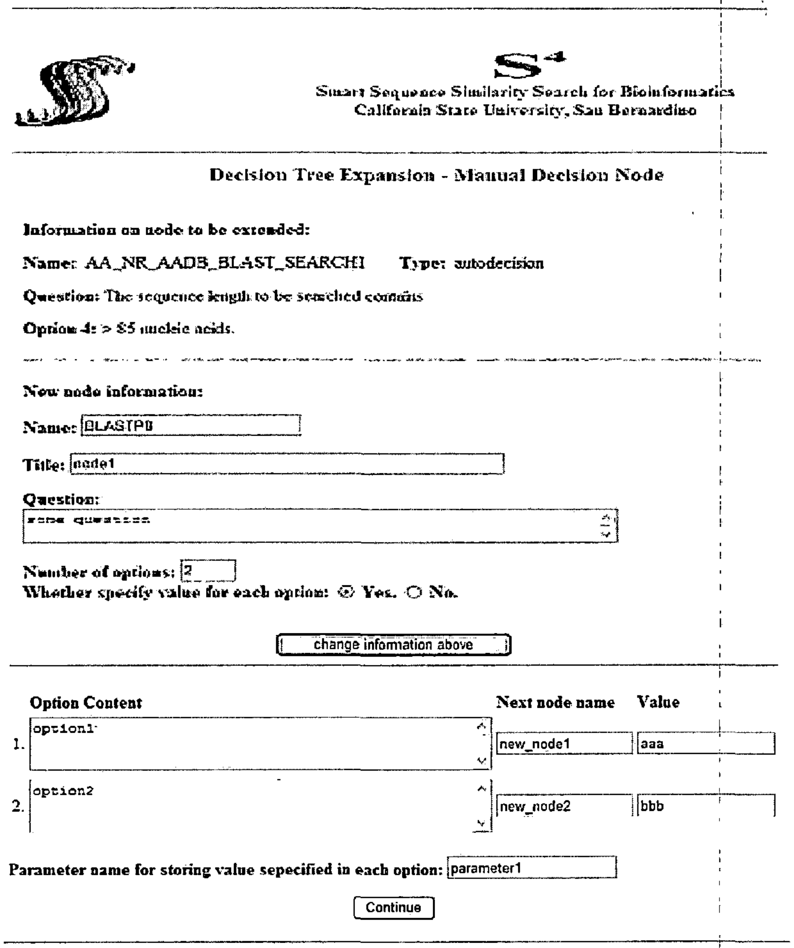
1. PROGRAM BLAST(BLASTP)
2. DATABASE nr
3. MATRIX_NAME BLOSUM62
4. WORD_SIZE 3
5. EXPECT 10
6. GAPCOSTS 11+1
7. ALIGNMENTS 50

Type of node to be extended in the Decision Tree

[✓ Manual Decision Node]
Auto Decision Node
Option Node
Leaf Node

Figure 2.8. Decision Tree Extension Page

<u>2.3.1.5.1 Input and Edit Page</u>. The page will allow the users to input the name, the title and the question to display (Figure 2.9). The options and associated parameters should also be provided by the users. The pages also allow the users to edit the changes again if necessary.


Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Decision Tree Expansion - Manual Decision Node

Information on node to be extended:

Name: AA_NR_AADB_BLAST_SEARCH1 **Type:** autodecision

Question: The sequence length to be searched contains

Option 4: > 85 nucleic acids.

New node information:

Name: BLASTP8

Title: node1

Question:
some question

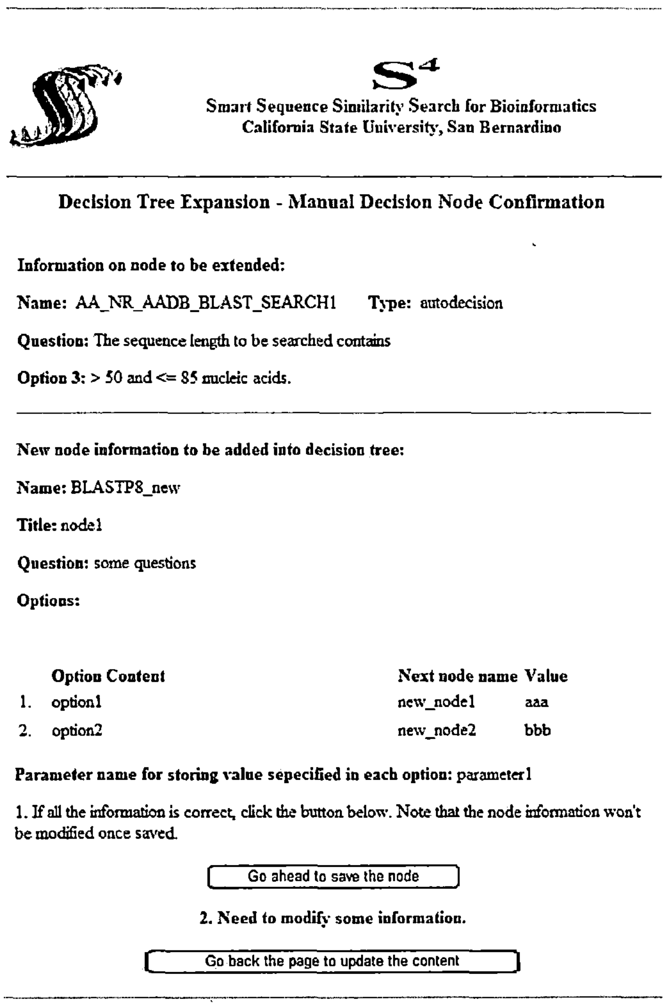
Number of options: 2
Whether specify value for each option: [✓ Yes.] No.

	Option Content	Next node name	Value
1.	option1	new_node1	aaa
2.	option2	new_node2	bbb

Parameter name for storing value sepecified in each option: parameter1

Figure 2.9. Manual Decision Node Extension Page

<u>2.3.1.5.2 Confirmation Page</u>. A confirmation page is provided for the users before the new information are saved temporarily (Figure 2.10). The page allows the user to go back to previous input page to modify the content. Once the information is submitted, the content can't be changed by the users any more.


S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Decision Tree Expansion - Manual Decision Node Confirmation

Information on node to be extended:

Name: AA_NR_AADB_BLAST_SEARCH1 **Type:** autodecision

Question: The sequence length to be searched contains

Option 3: > 50 and <= 85 nucleic acids.

New node information to be added into decision tree:

Name: BLASTP8_new

Title: node1

Question: some questions

Options:

	Option Content	Next node name	Value
1.	option1	new_node1	aaa
2.	option2	new_node2	bbb

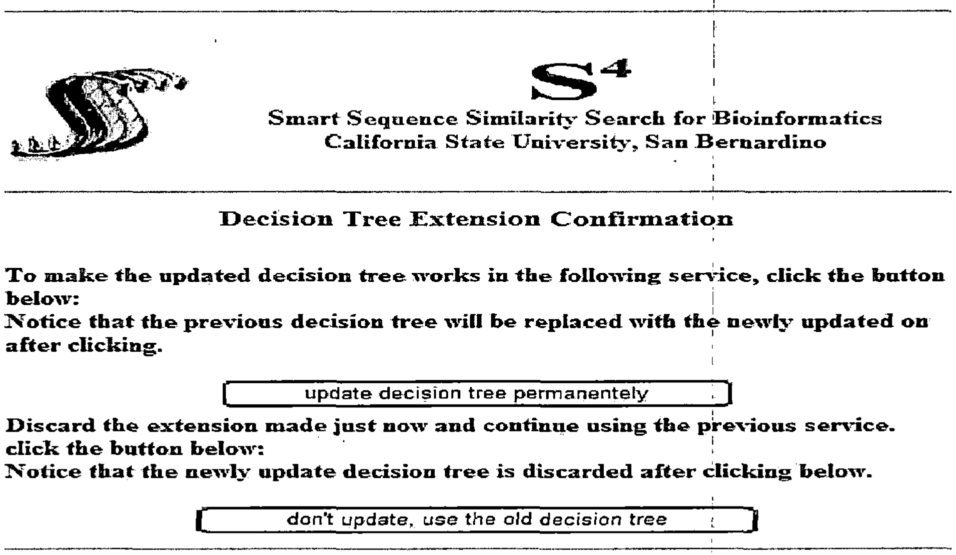
Parameter name for storing value sepecified in each option: parameter1

1. If all the information is correct, click the button below. Note that the node information won't be modified once saved.

2. Need to modify some information.

Figure 2.10. Manual Decision Node Extension Confirmation Page

2.3.1.5.3 Knowledge Update Confirmation Page. This page allows the user to update the system knowledge permanently or discard the new knowledge (Figure 2.11). If the change is saved, all the searches conducted later will be based on the updated knowledge.


S4

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Decision Tree Extension Confirmation

To make the updated decision tree works in the following service, click the button below:
Notice that the previous decision tree will be replaced with the newly updated on after clicking.

Discard the extension made just now and continue using the previous service.
click the button below:
Notice that the newly update decision tree is discarded after clicking below.

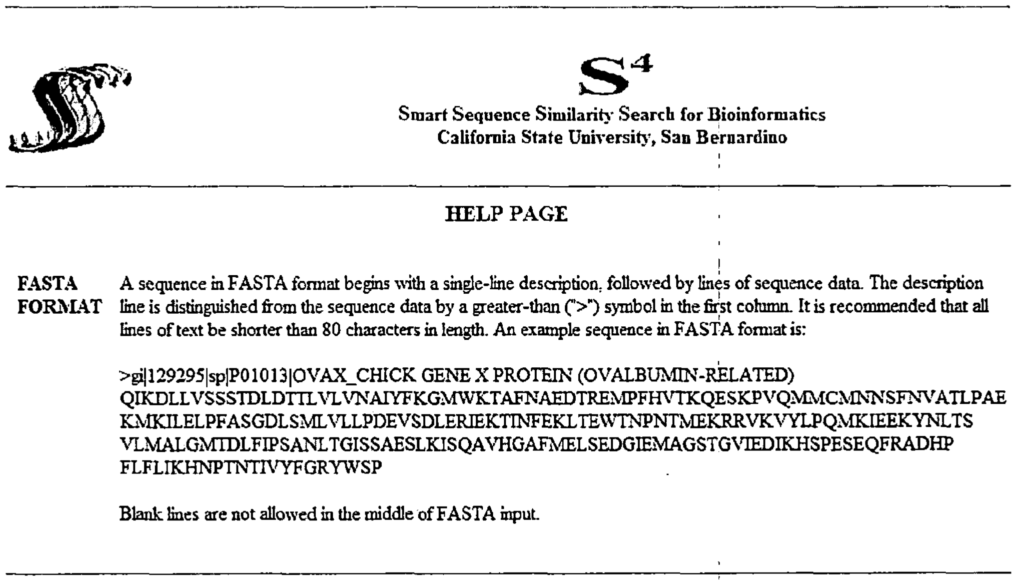
Figure 2.11. Decision Extension Confirmation Page

<u>2.3.1.6 Help Pages</u>. These pages provide users with introductory knowledge on sequence similarity search. These help pages can be accessed through links from other pages (Figure 2.12).

2.3.2 Functional Requirements

<u>2.3.2.1 Optimal Sequence Similarity Search program selection</u>. S4 system is designated to facilitate beginner to conduct his/her sequence similarity search properly.

The final program and corresponding parameter setting is determined by the system based on a series of options the user is provided to choose. The rules for making this final decision are initially based on the knowledge provided on the remote server help document and related researches conducted so far. Thus, it is not 100%


S4

Smart Sequence Similarity Search for Bioinformatics

California State University, San Bernardino

HELP PAGE

FASTA FORMAT A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. It is recommended that all lines of text be shorter than 80 characters in length. An example sequence in FASTA format is:

```
>gi|129295|sp|P01013|OVAX_CHICK GENE X PROTEIN (OVALBUMIN-RELATED)
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAE
KMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTS
VLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHP
FLFLIKHNPTNTIVYFGRYWSP
```

Blank lines are not allowed in the middle of FASTA input.

Figure 2.12. Help Page

guarantee that the program and its parameter chosen by the program are optimal. With more research results coming up, the rules would be modified later for better guidance.

2.3.3 Performance Requirements

S^4 will support approximately all researchers in CSUSB to use simultaneously. The response time to view any page should be less than one second when accessing from school or three seconds when accessing from home. The response time for seeing any result after submitting an input usually should not exceed twice the length than viewing a page.

2.3.4 Software System Attributes

2.3.4.1 Reliability. All contents and logs shall be generated dynamically and automatically so no human interference is needed. The sever shall be up twenty-four hours a day and seven days a week, with exception that periodical system maintenance needs to be conducted depending on the reliability of the web sever. The system should handle network packet loss smoothly. The system should not save inconsistent data or incomplete data into the system.

2.3.4.2 Security. The web-server used to execute the S^4 is open to public. No security issue is considered in this thread.

2.3.4.3 Maintainability. The system consists of java classes, JESS expert system and JSP pages. They will be put under different directories with hierarchy. Other files including java source code and documents will be put in separate directories as well. Java classes are organized using packages. This structure will aid in maintaining all modules organized and therefore maximizing maintenance facility.

2.3.4.4 Portability. The Server-Implementation of S^4 system shall be 100% portable since it will be written in Java, a proven portable language. The only determinant of how easily the S^4 is ported from on architecture to another is having the latest version of the Java Virtual Machine installed on the web-server machine. The Client portion of the S^4 will also be 100% portable since the system will be in presented using dynamic HTML pages and style sheet, which is supported by most up-to-date web browsers.

CHAPTER THREE

DESIGN OF S^4 SYSTEM

3.1 System Design

The S^4 system is an expert system, which use Jess as the expert system shell for the S4 project.

3.1.1 System Overview

Expert system is a computer system that emulates the decision-making ability of a human expert. It contains knowledge derived from an expert in some specific domains. This knowledge is presented in the form of fact(s) or rule(s) to help the user to solve various problems. The user can get advice from an expert system after supplying necessary information to it. It has been applied virtually to every field of knowledge, such as research tools to programs used in business and industry. Since it is primarily designed for symbolic reasoning, it is regarded as a suitable decision-making system in S^4 to help biologists choose suitable programs and parameters during sequence similarity search.

Jess is an expert system shell implemented in Java by Ernest Friedman-Hill in Sandia National Laboratories Livermore, CA. It was originally implemented based on the CLIPS expert system shell with many more features than

CLIPS, including backwards chaining, working memory queries, and the ability to manipulate and directly reason about Java objects. Besides, it provides a powerful Java scripting environment, from which we can create Java objects and call Java methods without compiling any Java code. It is regarded as an ideal expert system shell that can be integrated into Java programs for the purpose of providing "reasoning" with knowledge, which is supplied in the form of declarative rules and facts.

Similar to other expert systems, S^4 consists of four basic elements: inference engine, knowledge base, database and user interface, see Figure 3.1.

3.1.2 Inference Engine

Inference Engine makes inferences by deciding which rules are satisfied by facts, prioritizes the satisfied rules and executes the rule with the highest priority. In this way, it can draw conclusion or give advises to the user. Jess expert system shell provides full functions of inference engine. It uses Rete algorithm to process rules, which is a very efficient mechanism for solving the difficult many-to-many matching problem. No more implementation or improvement is required here.

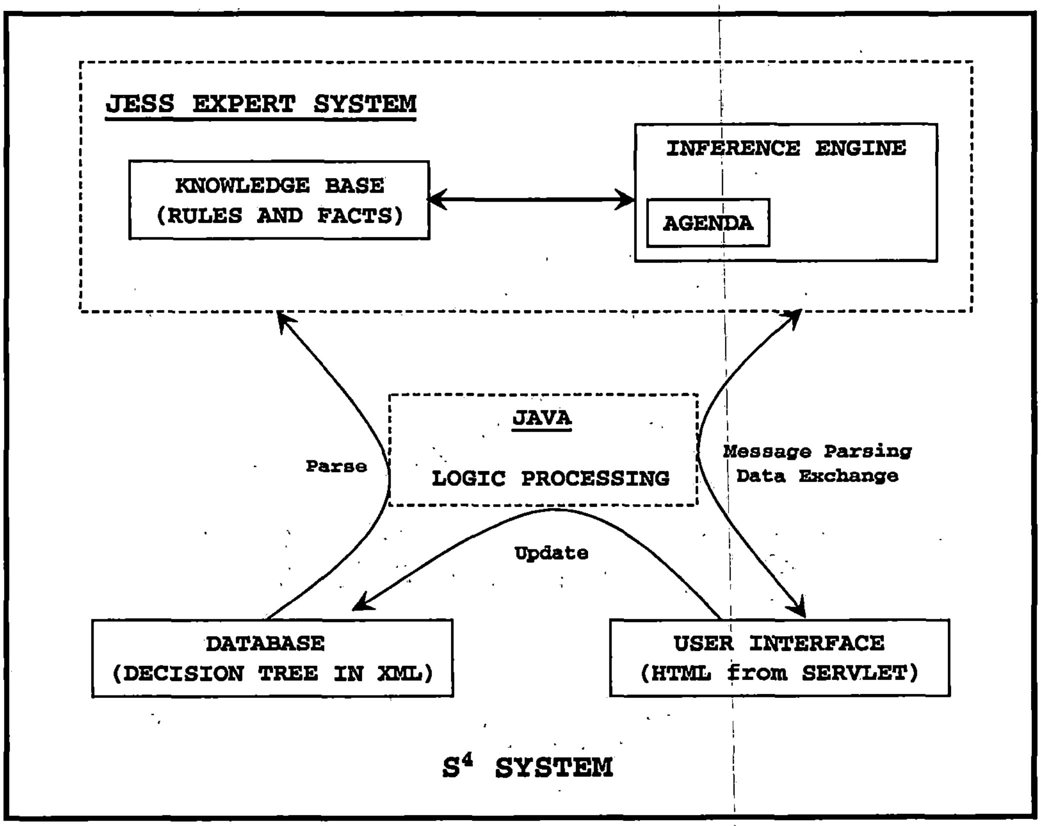

JESS EXPERT SYSTEM
KNOWLEDGE BASE (RULES AND FACTS)
INFERENCE ENGINE
AGENDA
JAVA
LOGIC PROCESSING
Parse
Message Parsing Data Exchange
Update
DATABASE (DECISION TREE IN XML)
USER INTERFACE (HTML from SERVLET)
S4 SYSTEM

Figure 3.1. S^4 Expert System Diagram

3.1.3 Knowledge Base

S^4 system contains the knowledge on sequence similarity search in the form of facts and rules. The knowledge is extracted from experts in this field or from the user manuals and tutorials of the various online sequence similarity search services considered in the system. This knowledge is organized and presented in the form of a decision tree in S^4 system.

3.1.3.1 Decision Tree. A decision tree possesses (typically) a single root, a set of branches from that root, interior nodes that also have branches and leaf nodes, which present the classification of information. Each node within the tree represents a decision point that determines which subsequent branch is followed until a leaf node is reached. The leaf nodes in the tree represent all the possible solutions that can be derived from the tree, which is also called answer node. Thus, a defined decision tree provides a user paradigm for solving certain type of classification problems, and finally providing the answer to a problem from a predetermined set of possible answers. This is exactly comparable to the objective of our project. Moreover, a tree can be easily extended by deleting a leaf node and replaced it with interior nodes such that each of these nodes will have open paths leading to newly defined leaf nodes. The extension allows new knowledge to be included in a decision tree, making it richer and more accurate as more information is inputted by users who are experts in the domain of sequence similarity search. This extensibility provides S^4 the ability to learn from expert users for the purpose of service improvement. The process of decision tree extension is also called decision learning.

3.1.3.2 Knowledge presentation method. S^4 expert system uses facts to represent knowledge stored in a decision tree, and rules to determine the process of decision tree moving. Because facts can be added or removed easily to update the tree as it learns, it is worthwhile to represent the tree as facts instead of rules in expert system, which is convenient for decision tree extension. Thus, each node of the decision tree will be represented by a fact with a unique name. The necessary content of an interior node in the decision tree started from a root node might be displayed to the user for selection, and the system will continue the process to next node until a leaf node is reached. This process is determined by rules written for decision tree traverse. All the facts representing the S^4 decision tree will be stored in a local file when the user exits S^4 system. They can be retrieved and asserted from the same file when the program starts next time.

3.1.3.4 Programming Language. Jess supports rule-based expert system development both in its specific CLIPS-like scripting language and Java. We used Java to support Web-based interface, communication and information exchange with other parts of the system. We chose Jess scripts for the knowledge base.

3.1.4 Database

A database is used to permanently store knowledge in the form of a decision tree. To facilitate knowledge construction, update and maintenance, the whole S^4 decision tree is initially presented in XML with the tree structure predefined in corresponding DTD file. The decision tree in XML can be read and asserted as facts into Jess system.

Due to its high readability, the content of the decision tree can be easily updated or extended directly on XML file using any text editor. Moreover, it can also be updated automatically by S^4 system when the tree is extended by the user.

3.1.5 User Interface

Jess provides powerful functions to support applet as the user interface. However, it is not considered as a good technique for Internet application because of the large amount of Jess data that has to be transferred via Internet. Besides, applets might limit some Java functions due to security issues. Since S^4 is designated to provide a Web-based service, we considered Java Server Page (JSP) pages and Servlets as programming techniques to support the communication between S^4 expert system and the users.

In detail, once the inference engine determines that a node in the decision tree should be displayed to the user for option selection, or a leaf node is reached, a JSP page containing the necessary information of the node will be automatically generated and sent to the user. In the meantime, the expert system should wait for the user's response submitted from JSP page before deciding which rule to fire. Thus, two threads are required in the S^4 system. One is for Jess rule engine for "reasoning" upon the initial request from the user, which will be continuously running until an "exit" command is received from the user. The other thread is responsible for automatically generating the content of the JSP page to be sent to the user and obtaining response from the user in HTML form. Communication between the two threads is achieved by pipes in order to guarantee the synchronization, while knowledge information exchange is fulfilled by Jess functions to store and retrieve variables from the Java application.

3.2 Decision Tree Structure Design

All the knowledge on S^4 is collected either from user manuals and tutorials of NCBI BLAST Service, EBI FASTA Service and DDBJ S&W Search Service or experts in this field. A decision tree is constructed manually based on the obtained knowledge.

3.2.1 Overall Structure

The S^4 decision tree consists of nodes and branches. The leaf nodes represent all the possible recommendations that the system can provide. They are also called answer nodes in Jess expert system code. All other nodes in the tree are referred to as decision nodes. To better represent the knowledge included in the S^4 decision tree, three types of decisions are constructed: manual decision node, auto decision node, and option node.

(a) Manual decision node: The location of the decision tree will move from this node to corresponding child node based on the branch the user selects.

(b) Auto decision node: The location of decision will move from this node to corresponding child node based on the branch automatically selected by the system using predefined branch-specific criterion.

(c) Option node: No matter which branch the user selects, there is only one child node that the decision tree will move to. It is responsible for setting the value for a specified parameter based on the user selection.

The structure of the S4 Decision tree is managed in a declarative form to facilitate information storage in XML format. As a result, we store information on each branch of decision node into its parent node as well as the corresponding child node name. Thus, the decision tree in XML format consists solely of three types of decision nodes and leaf nodes.

Nodes in XML format basically follow the hierarchical structure design. The detail structure of each type of nodes is defined in a Document Type Definition (DTD) file and illustrated one by one in the following tree diagram. In each tree diagram, rectangle box represents an element in the decision tree, while ellipse box represents an attribute of an element it belongs to. Labels of the edges represent the number of attributes or elements the parent elements should contain.

3.2.2 Structure of Decision Node

The three decision nodes share common structures and each also contains its unique structure. The common structures are two attributes (name and title) and two elements (question and help). Name is the unique name of the node. Title defines the title of the page when the node content is displayed to the user. Question element contains the question that will be asked to the user. Help element contains the ID or jargon that can link to the separate help page. Unique structures for each decision node will be discussed in the following sections.

3.2.2.1 Manual Decision Node. This type of node can contain one or more branch elements (Figure 3.2). Each branch element must have one content attribute and one child name attribute. Content attribute contains the option displayed to the user, and child name is the name for its corresponding child node to the option. If there is a parameter element, the value attribute of the branch element must also be set. There is only a name attribute for the parameter element.

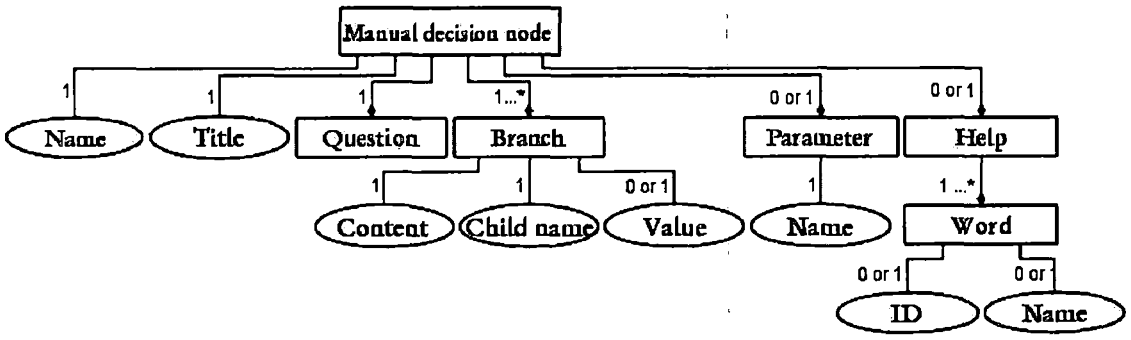


Figure 3.2. Tree Structure of Manual Decision Node in XML

3.2.2.2 Auto Decision Node. This node contains a number of branch_rule elements (Figure 3.3). Each branch_rule element has one branch element and one or more rule elements. The branch node here is the same as in manual decision node. The parameter element also determines whether the value attribute in the branch element should be set. Each rule element contains three attributes, parameter, operator and value. Parameter and value attributes were compared based on the operator to determine whether this branch should be selected by the system.

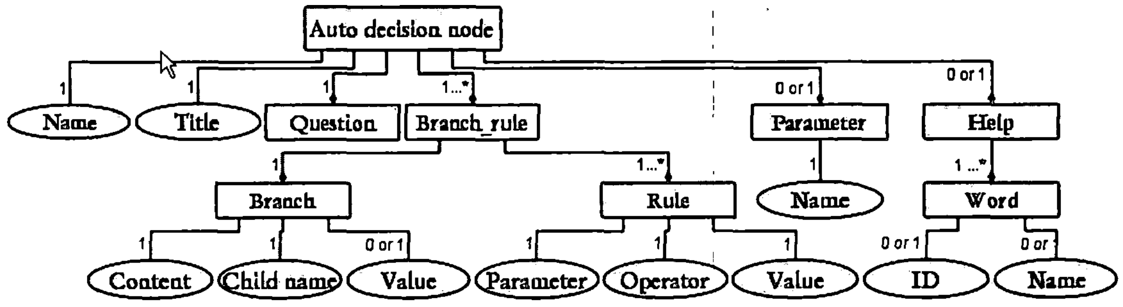


Figure 3.3. Tree Structure of Auto Decision Node in XML

<u>3.2.2.3 Option Node</u>: This type of node contains one or more option elements (Figure 3.4). The explanation attribute of each option element contains the information that will be displayed to the user. The value attribute of each option element determines the value of the parameter whose name is specified by the parameter element. The child element specifies the only child node that the location where decision tree will move to.

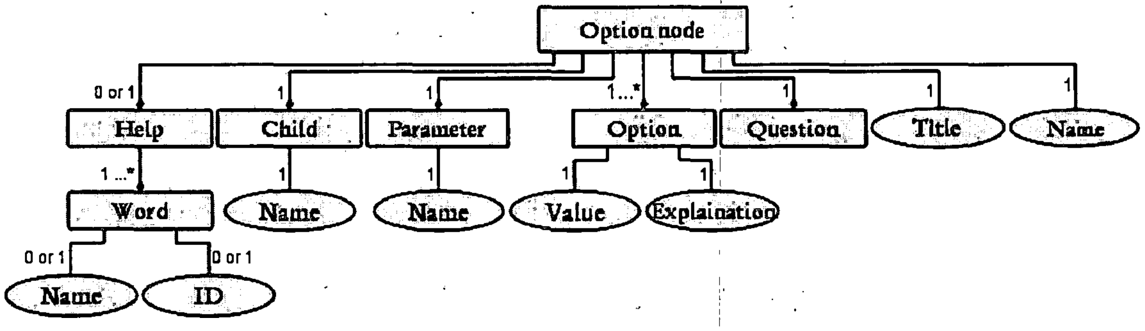


Figure 3.4. Tree Structure of Option Node in XML

3.2.3 Structure of Leaf Nodes

As shown in Figure 3.5, leaf node contains a name attribute which defines the name of the node, one help element and one or more parameter elements. Each parameter element contains a name attribute and a value attribute. The help element here is the same as that in decision node.

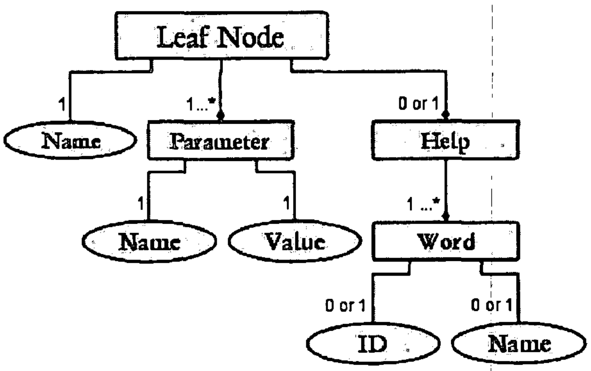


Figure 3.5. Tree Structure of Leaf Node in XML

3.3 Architecture

S4 system is developed in Java to facilitate interface communication with Jess expert system and support online service using Servlet and JSP.

3.3.1 Overview

As shown in Figure 3.6, the whole system consists of four subpackages (system, xml, bean and utility) under package csusb.s4. Among these, system is the core subpackage and directly responsible for initializing Jess expert system, providing Web-based user interfaces and facilitating communication between user and expert system. Bean subpackage contains Java bean classes used to facilitate decision tree information storage and retrieval. XML subpackage contains functions to communicate with the XML file in order to retrieve information from it or update its content. Utility subpackage is an auxiliary package with reusable functions in the system. We will discuss the first three subpackages in the following sections.

3.3.2 Package csusb.s4.system

As shown in Figure 3.7, system package contains five classes: S4ExpertDecisionServlet, S4JessExpert, S4ExpertLearner, IOExchanger and S4Exception.

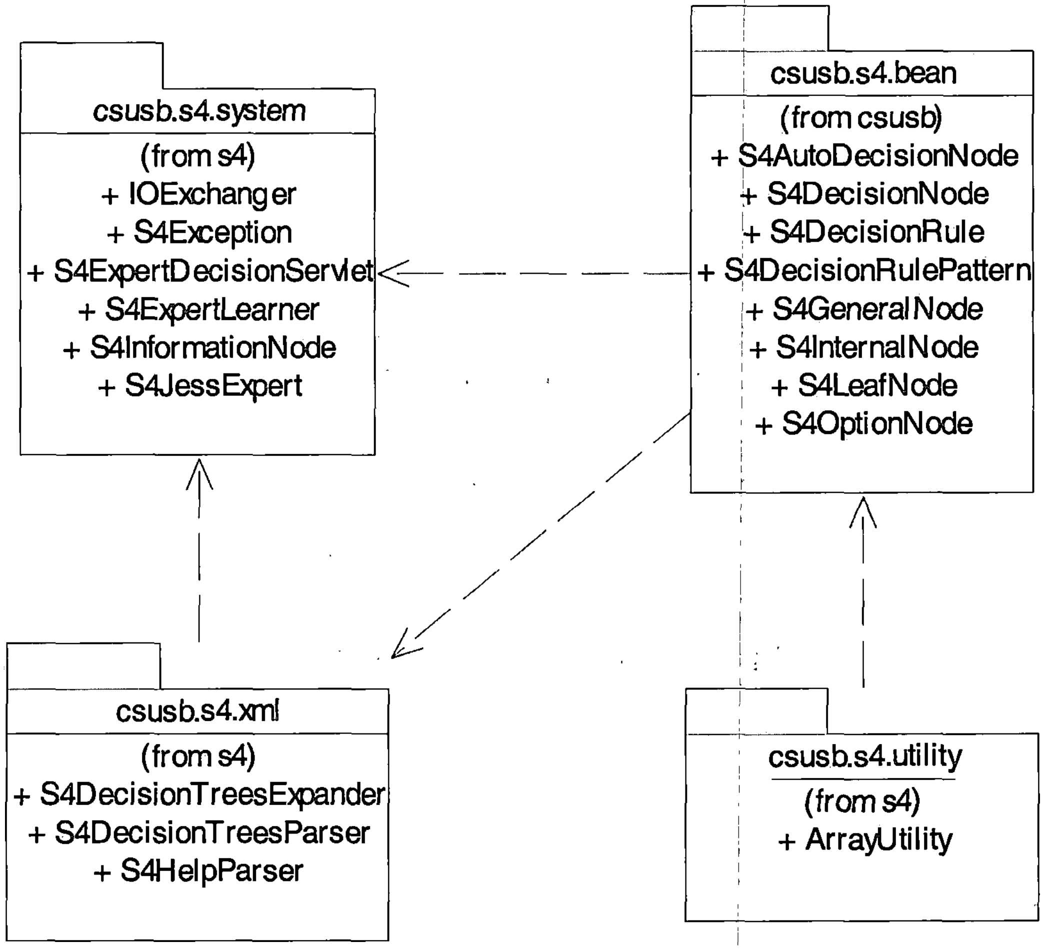


Figure 3.6. Subpackage Diagram of csusb.s4 package

<u>3.3.2.1 Class S4ExpertDecisonServlet</u>.

S4ExpertDecisionServlet is a servlet class responsible for communication between the user and the Jess expert system. Upon receiving initial request from the user, it creates a new thread S4JessExpert and sends request to Jess with necessary information for the expert system operation. After that, it communicates with the Jess expert system after receiving the user response and creates HTML page to

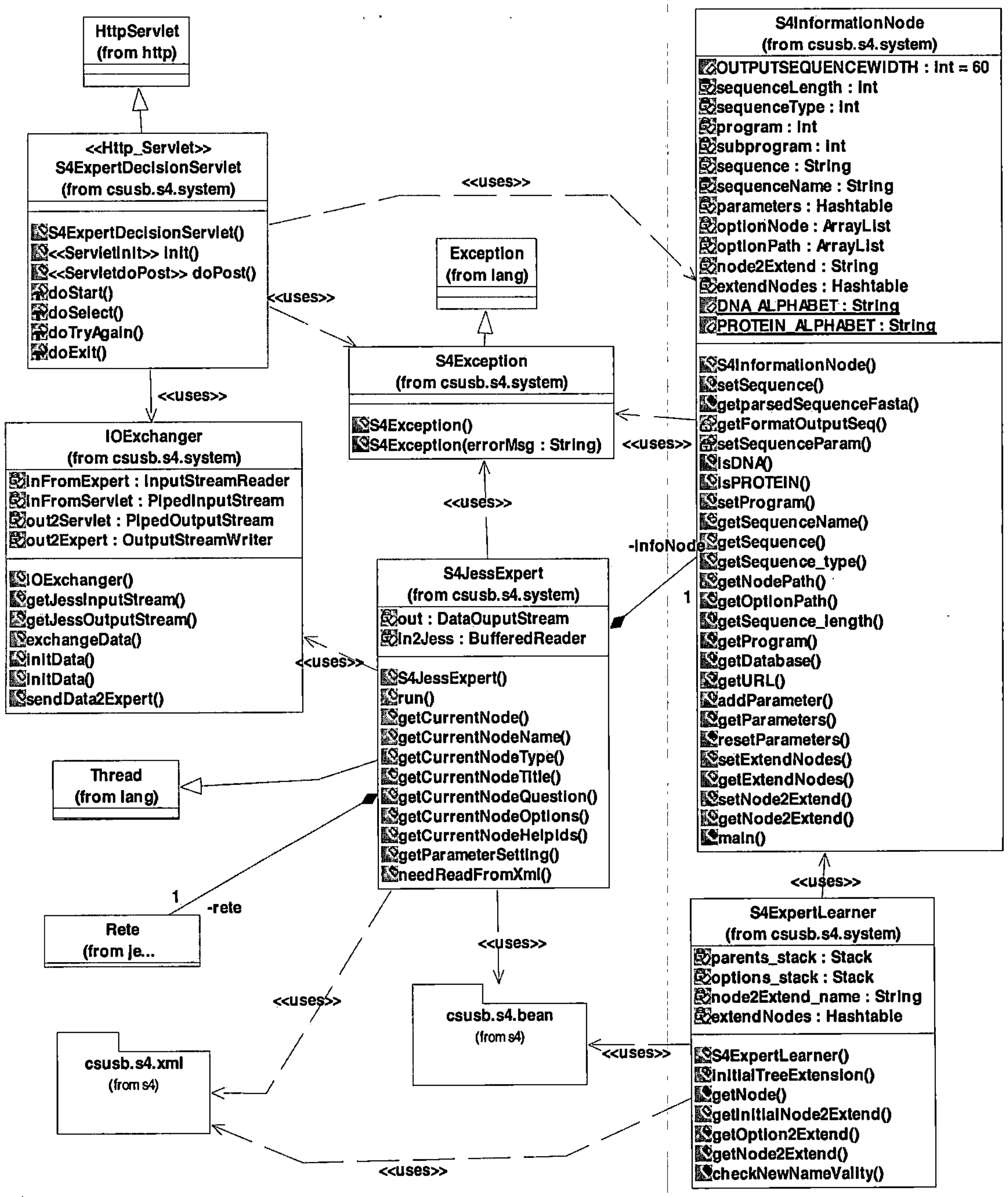


Figure 3.7. Class Diagram of Package csusb.s4.system

display information obtained from Jess to the user. It is also responsible for sending user's selection or command as message to Jess to direct its further running.

3.3.2.2 Class S4JessExpert. S4JessExpert extends Thread class so that it starts a different thread from S4ExpertDecisionServlet. It is the thread on which the Jess expert system runs. Upon starting Jess, it establishes the knowledge base coming from two sources. One is the files containing Jess facts and rules and do not need to be parsed. The other is the XML file storing the decision tree, and needs to be parsed to generate decision node objects. Please see section 3.4.1.1 for a detailed knowledge base construction.

Then, the processing for decision making is completely based on Jess rules and will be further discussed in Section 3.4.1.2. The thread will continue running with pauses for user commands or selections until an "exit" command is received from the servlet. When Jess is paused because of a user command, the inner class EventHandler implementing EventListner is responsible for notifying the parent thread, which, in turn, communicates with the user to get proper command and send it to Jess.

The communication between servlet and Jess is supported by IOExchange class, which provides two communication pipes and related functions to facilitate the two-direction communication. This communication is not only used to transfer message between the threads, but also to guarantee the synchronization of the two threads.

3.3.2.3 Class S4ExpertLearner. S4ExpertLearner provides functions for the users to extend the decision tree through either updating the XML file or the knowledge base of the current running Jess engine. Updating the XML file provides permanent information storage and mainly utilizes the functions provided by S4DecisionExpander object in Package csusb.s4.xml. The updated XML file will be parsed for later usage. Updating the Jess knowledge base immediately takes effect and is convenient for the users who want to use the updated knowledge right away. S4ExpertLearner is responsible for this extension process. Detailed algorithm is provided in Section 3.4.2.

3.3.2.4 Class S4InformationNode. S4InformationNode is the class for storing the useful information and suggestions generated during the processing of expert system operation. The object can be accessed by Jess engine thread and the thread handling user interface. Thus, it supports information exchange between threads running on the same machine without data transfer.

3.3.2.5 Class S4Exception. S4Exception extends Exception class and is used to handle the exception thrown in the system application.

3.3.3 Package csusb.s4.bean

Bean package contains six bean classes and two related classes. All these classes are used to store information of the four types of nodes in the decision tree, as discussing in Section 3.2, and their corresponding functions. Designed as bean classes, the objects can not only be easily asserted into Jess engine as instances, but also facilitate the information retrieval and display in JSP forms. They can also be constructed by and utilized in XML parser.

As shown in Figure 3.8, the classes for four types of nodes are designed as a 4-layer inheritance. Among them, S4GeneralNode is the super class and the common structures of all four types of the nodes. S4LeafNode and S4InternalNode directly extend S4GeneralNode. S4LeafNode is specific for leaf node information, while S4InternalNode represents all the common features of interior nodes. S4InternalNode is the super class of S4DecisionNode and S4OptionNode, which represents manual decision node and option node respectively. Auto decision node is designed as S4AutoDecisionNode which extends

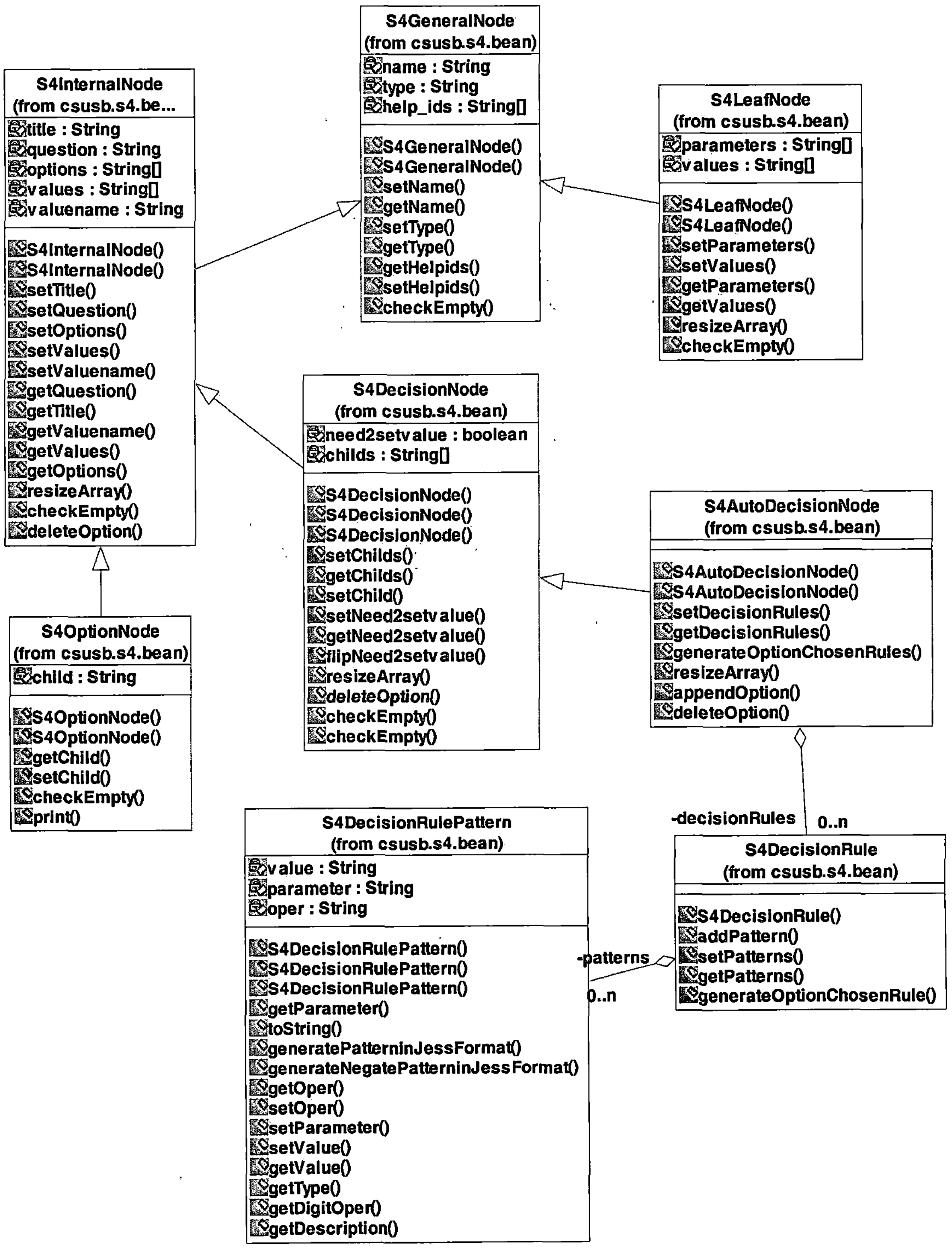


Figure 3.8. Class Diagram of Package csusb.s4.bean

S4DecisionNode with additional information on decision rules.

Each auto decision branch contains a decision rule, which uses S4DecisionRule containing its properties and related functions. Each rule object will further contain one to several patterns as S4DecisionRulePattern object, which is the basic unit for the rule to be constructed.

3.3.4 Package csusb.s4.xml

Xml package contains the classes that are responsible for parsing XML file to retrieve information or updating the content of XML file, see Figure 3.9. Document Object Model (DOM) is used in all the classes to take advantage of its available functions supporting XML update.

S4DecisionTreeParser class is used for retrieving decision tree information from XML file and constructing lists of bean objects with different node types. The constructed objects can be subsequently asserted into Jess as instances, as mentioned before in Section 3.3.2.

S4DecisionTreesParser is the class providing functions for XML file update, and mainly used by S4DecisionLearner object in csusb.s4.system package. S4HelpParser is the class specific for parsing help page content from XML files as well as constructing look up table for help page display.

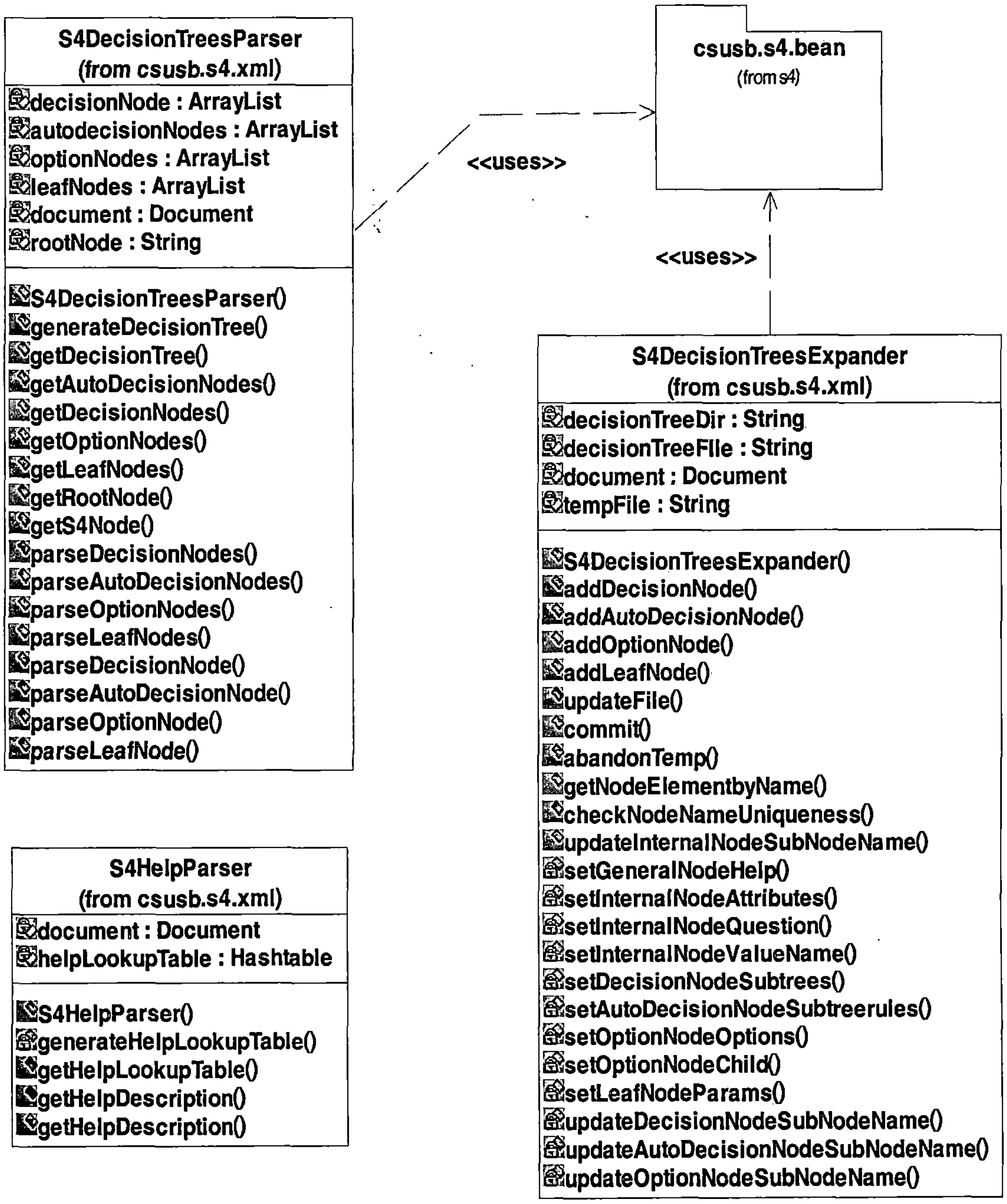


Figure 3.9. Class Diagram of Package csusb.s4.xml

3.4 Detailed Design

This section contains the detailed design for the expert knowledge construction, processing and decision tree extension method.

3.4.1 Jess Expert System

Jess expert system is the logic processing core of the whole application. Using the comprehensive knowledge on S^4 decision tree, its Rete engine will continue running to communicate with the user until an "exit" command is given by the user. Rules are defined and fired to support proper running of the engine.

To improve performance, these rules are classified into four modules, MAIN, CONTROL, DECISION-MAKING and DECISION-UPDATE. MAIN module is the initial module running and is used to determine which of the other three modules should be executed. CONTROL module is responsible for getting feedback from the users after suggestions are given to the user. DECISION-MAKING module is the major module to import decision tree information to the engine and determine how the decision tree is traversed from the root node to an answer node. Detailed algorithm will be shown in Section 3.4.1.1. DECISION-UPDATE module contains rules responsible for decision tree extension in Jess engine, which will be discussed in section 3.4.1.2. Once the state of the engine satisfies a rule, it will be placed on the agenda of the engine. The rules in the agenda will be fired one by one using "breadth" strategy.

3.4.1.1 Decision Tree Knowledge Base Construction. Expert knowledge data on similarity search in decision tree is mainly presented in the form of facts except the rules generated for auto decision tree option selection.

The facts' templates for the knowledge are defined using Jess "defclass" function. In this way, the slots in each template representing a specific type of node are named after the properties of the corresponding bean class. The fact templates' dependencies also remain the same as those of the bean classes using "extends" option. As a result, the decision making facts' structure is consistent with Java bean classes' architecture as shown in package csusb.s4.bean. Jess can then use these templates to store knowledge representations by two strategies. One is from Java bean objects by using function "definstance". The objects are normally generated by parsing the decision tree in XML file. The other is from previously saved text file by Jess in the form of facts using Jess function "load-facts".

The whole process of constructing the knowledge base uses both Java and Jess programming languages. Within the process, the system compares the last modified time of the XML file and Jess-saved file to determine which knowledge source should be used. This is to speed up the process of knowledge construction while guaranteeing the most updated knowledge is used. See Listing 3.1 and 3.2 for pseudo-code algorithm of the process as implemented in Java and Jess languages respectively.

While the knowledge from the decision tree is represented in the form of facts, another form of knowledge is the auto decision node's option which is represented in the form of rules. These rules are generated by Java from rule-related information stored in S4DecisionTreeRule and S4DecisionRulePattern objects. Each option of an auto decision node has a rule to determine its selection criterion. The algorithm for the rule generation is shown in Listing 3.3. All the generated rules are stored in a text file which can be retrieved into Jess using "batch" Jess function.

```
if needReadFromXml() is true:
   create S4DecisionTreesParser object dtParser;
   parse decision tree in XML by function
generateDecisionTree()
   set decisionNodes to a list of manual decision nodes
parsed
   set autodecisionNodes to a list of auto decision nodes
parsed
   set optionNodes to a list of option nodes parsed
   set leafNodes to a list of leaf nodes parsed
   put variables of four lists of nodes into Jess
   batch execute the text file containing Jess rule for
reasoning
   open a new FileWriter pw;
   for each autodecisionNode object:
      generate option-selection rules;
      build rules into Jess;
      write rules to pw
   end
   close pw
   assert fact into Jess to indicate using xml file to
start
else:
   batch execute the text file containing Jess rule for
reasoning
   batch execute the text file containing Jess rules for
autodecision nodes' option-selection
   assert fact into Jess to indicate using Jess fact file
to start
endif
start running Jess engine

Function Boolean needReadFromXml() {
   Set xmlModifiedTime to the last modified time of xml
File;
   Set factModifiedTime to the last modified time of jess-
save   text file containing facts.
   Set ruleModifiedTime to the last modified time of Java
generated text file containing auto decision tree rules.
   if (xmlModifiedTime later than factModifiedTime
      Or xmlModifiedTime later than ruleModifiedTime)
      return true;
   else
      return false;
  }
```

Listing 3.1. Pseudo Code of Jess Knowledge Base

Construction (Java Part)

```
In Module DECISION-MAKING, define following template
in order:
  generalnode by S4GeneralNode class
  answernode by S4LeafNode class extends generalnode
  internalNode by S4InternalNode class extends
generalnode
  optionnode  by S4OptionNode class extends
internalnode
  decisionnode by S4DecisionNode class extends
internalnode
  autodecisionnode by S4AutoDecisionNode class extends
decisionnode

 RULE 1: Set up decision tree information from xml
 file
     IF    Information is parsed from xml file
     THEN  1) Fetch four lists of nodes objects;
           2) Assert all the manual decision node
           objects as decisionnode fact one by one;
           3) Assert all the auto decision node
           objects as autodecisionnode fact one by
           one;
           4) Assert all the option node objects as
           optionnode fact one by one;
           5) Assert all the leaf node objects as
           optionnode fact one by one;
           6) Assert initial parameter information
           from Java application as facts.

 RULE 2: Set up decision tree information from Jess
 stored file
    IF    Information is obtained from files for facts
          and rules.
    THEN  1) Load text file containing fact in four
          node types' templates;
          2) Assert initial parameter information from
          Java application as facts.
```

Listing 3.2. Pseudo Code of Jess Knowledge Base's

Construction (Jess Part)

```
Class S4DecisionRule:
   Properties: A list of patterns as S4DecisionRulePattern objects
   Function String generateOptionChosenRule (nodename,
optionnumber) {
       initialize variable rule in format
         (defrule DECISION-MAKING::choose_<nodename>_<optionnumber>
             ?choose <- (auto_choose_option <nodename>)
       for each pattern in the rule
          append pattern.generatePatternInJessFormat()
       end
       close rule with
             =>
             (retract ?choose)
             (assert (option <optionnumber>)))
        return rule
    }
endclass

Class S4DecisionRulePattern:
   Properties: Parameter name, type, value and operator
   Function String generatePatternInJessFormat() {
      Set variable pattern with function toString()
      Generate pattern machine fact in format
         (parameter "<parameter name>" ?x&: <pattern>)
      return generated fact as String
   }
   Function String toString() {
      Declare String Variable pattern;
      if (parameter type is number)
         set variable pattern in format (<operator> ?x <value>)
      else if (parameter type is String)
         if (operator is EQUALS or LARGER THAN or LESS THAN)
            set pattern in format (=<operator> (str-compare ?x
"<value>")
         else
            initialize pattern with (or
            if (operator contains "EQUALS")
               append pattern with (= 1 (str-compare ?x "<value>"))
            if (operator contains "LARGER THAN")
               append pattern with (= 2 (str-compare ?x "<value>"))
            if (operator contains "LESS THAN")
               append pattern with (= 4 (str-compare ?x "<value>"))
            append pattern with )
          endif
      endif
      return pattern
   }
endclass
```

Listing 3.3. Pseudo Code of Jess Rule Generation for Auto-Decision Node's Option

3.4.1.2 Rules for decision tree knowledge processing.

The rules to process decision tree in module DECISION-MAKING after finishing knowledge construction is shown in Listing 3.4.

```
RULE 1: Output current interior node information
   IF     1) Current node is newly set without further processing
          2) Current node is interior node
   THEN   Retrieve name, type, title, question, options and help
          word id and set to corresponding variables.
RULE 2: Output current leaf node information
   IF     Current node is leaf node.
   THEN   Retrieve name, type and set to corresponding variables.
RULE 3: Determine auto decision node option.
   IF     1)Current node is auto decision node;
          2) No option has been selected by Jess.
   THEN   Check auto decision node option rules for option
          selection.
RULE 4:Option selection for manual decision or option node
   IF     1) Current node is manual decision node or option node;
          2) User has not given the selection.
   THEN   1) Display information to user;
          2) Ask user to choose one option.
RULE 5: Ask user's option when the node is auto decision node
   IF     1) Current node is auto decision node;
          2) Jess could not determine the option based on rules.
   THEN   1) Display information to user;
          2) Ask user to choose one option.
RULE 6: Tree proceeding for decision-node with param to be
set
   IF     1) Current node is decision node;
          2) One parameter need to be set based on selection.
          3) An option is selected by user or Jess.
   THEN   1) Set current node to the child node of the option;
          2) Set parameter value for the option selected.
RULE 7: Tree proceeding for decision node without param to be
set
   IF     1) Current node is decision node;
          2) No parameter need to be set based on selection.
          3) An option is selected by user or Jess.
RULE 8: Tree proceeding for option node
   IF     1) Current node is option node;
          2) An option is selected by user.
   THEN   1) Set current node to the child node;
          2) Set parameter value for the option selected.
```

```
RULE 9: Reach leaf node
   IF     Current node is answer node
   THEN   1) Set parameters value specified in answer node.
          2) Return to MAIN module.
```

Listing 3.4. Pseudo Code of Decision Tree Processing Rules

3.4.1.3 Rules for decision tree extension. The rules to process decision tree extension in module DECISION-UPDATE is listed as pseudo code in listing 3.5.

RULE 1: Initialize decision tree extension		
	IF	Time to extend tree
	THEN	Fetch node name to be extended from Java application
RULE 2: Delete leaf node and node to be extend from facts list		
	IF	1) An interior node to be extended is found;
		2) A leaf node as current node is found.
	THEN	1) Delete interior node to extended;
		2) Delete leaf node
RULE 3: Extend current decision tree		
	IF	Time to add new nodes
	THEN	1) Fetch all the new nodes including the node to be extended with updated child node name;
		2) Add the nodes one by one into Jess as facts using corresponding fact template into DECISION-MAKING module;
		3) Save nodes' information as facts in a flat file;
		4) Return to MAIN module.

Listing 3.5. Pseudo Code of Jess Decision Tree Extension Rules

3.4.2 Decision Tree Extension Procedure

S^4 system provides functions to allow the users to extend the decision tree from a given leaf node. First, the currently located leaf node in the application is deleted and the extension is conducted on the subsequently opened branch of the parent node. A subtree which might contain any type of the four nodes can be added to the open path after the leaf node is deleted. Then each node in the new subtree is added one by one in the depth-first order until all the newly-created open branches are closed with a leaf node. sS4ExpertLearner provides functions to guarantee that these new nodes are added in depth-first order using two stacks to trace the tree's location. It also provides functions to store the information temporarily and update the information permanently. The pseudo-code algorithm, which is implemented in Java, is shown Listing 3.6, while the part that is implemented in Jess is already mentioned in section 3.4.1.4.

```
Class S4ExpertLearner
   Properties:
      treeExpander: S4DecisionTreeExpander object for xml file
update
      parents_stack: stack of names of interior nodes to be
extended
      options_stack: stack of option numbers to be extended.
      extendNodes: newly-added node objects
   Function initialTreeExtension(S4InformationNode infoNode) {
     Get tree traverse path information from infoNode which is set
by Jess
     Call treeExpander to delete leafNode named by the last node
in the path
     Push second last node name in the path to parents_stack
     Push the branch number of node from which to be extend to
options_stack
     Add parent node into extendNodes as S4GeneralNode object
   }
   Function addNewTempNode(S4GeneralNode node) {
      Update parent node's corresponding child name with
node.getName()
```

```
      Add node to temp XML file by treeExpander and update the
file
      Add node to extendNodes list.
      Pop out last name of options_stack
      if (node is S4DecionNode)
         if (node is S4AutoDecisionNode)
            add auto selection rules generated for each option to
temp rule file. Push number 0 to options_stack
         Push integer max_option_number to 1 in order into
option_stack
         Push node name to parents_stack;
      else if (node is S4OptionNode)
          Push number 0 and 1 to options_stack in order.
          Push node name to parents_stack;
      End
   }
   Function Boolean time2ConfirmExtension() {
      if (options_stack is empty)
         return true;
      while (last element in options_stack is 0)
         pop out stack once;
         if (options_stack is empty)
            return true;
         parents_stack.pop();
       }
       return false;
   }
   Function updateTreePermernently() {
      Replace original decision tree XML file with temp XML file
by treeExpander
      Set node2Extend list to be accessed by Jess
      Send "extend" command to Jess by IOExchange object
   }
Endclass
```

Listing 3.6. Pseudo Code of Decision Tree Extension Procedure Control

3.4.3 Choices of Decision Tree

S^4 system provides two decision trees for the users to choose. One is an almost empty decision tree. It is used as an initial template for the users to populate the decision tree as the way he/she wants using decision tree extension interfaces. It is mainly provided to the expert users who had extensive knowledge on sequence similarity search and have their own desired methods to assist other users doing similarity search. In this almost empty decision tree, there are only three nodes - an auto decision node for determining the sequence type and two leaf nodes (one for protein sequence and the other for nucleic acid sequence) that serve as the point for decision tree extension. The other one is a populated decision tree constructed by S^4 system. The knowledge in this decision tree is mainly collected from literatures including user manuals or tutorials by the remote sequence similarity search services, and is further verified by experts in this field. The users can choose it in order to get assistance while doing their sequence similarity search. In the next chapter, we will discuss the knowledge included in this decision tree.

CHAPTER FOUR

POPULATING THE DECISION TREE

4.1 Introduction

The expert knowledge of sequence similarity search is stored in a decision tree. The construction of the decision tree is based on the available subprograms, databases, and functions of the three similarity search servers as well as their performance.

Whether the decision tree branches at certain point is carefully determined to make the optimal suggestion as unique as possible. Each search option results in a unique path through the decision tree and the trajectory mimic the decision-making process by a real researcher. In the real world, a research needs to decide his/her options before conducting the real sequence similarity search:

1. What is the goal of the search, e.g. to identify the sequence or to find similar sequences.
2. What type of sequence is the query? What database type is searched against, e.g. a DNA sequence containing coding region is searched against protein database.
3. What databases to search? For example, comprehensive databases, species specific

database, most updated database, or raw sequence database.

4. What is the parameter settings based on the characteristics of the query? For example, its length.

5. Which matters the most for the results? Speed, sensitivity, or selectivity.

Our decision tree encompasses all of the above information to mimic the search process of a real researcher. The decision tree diverges first at the subprogram level based on query sequence and purpose of the search. The next level is the databases and program-specific functions, which is followed by the sequence characteristics. The last level is the program performance. Divergence at different levels allows our decision tree to handle most of the commonly-used search options. Because of the large number of possible search options, there are over 400 nodes in the populated decision tree. In Sections 4.2 - 4.6, each of the search options implemented in the decision tree will be discussed.

4.2 Subprograms

As shown in Table 4.1, three commonly-used sequence similarity search servers provide different subprograms for searches with different types of query sequences and database sequences. All these subprograms are associated with a set of default parameter settings. Some servers provide unique programs to handle special search cases. For example, FASTA does not provide function to compare translated DNA query vs. translated DNA databases, but is the only one to provide functions for search short peptide patterns against protein database, which is particularly useful for identifying peptides from mass spectrometry or protease cleavage. FASTX3 and FASTY3 are designed to handle DNA sequences that might contain errors, which are commonly seen in EST sequences and sequences obtained from high throughput sequencing projects. MEGABLAST and discontiguous MEGABLAST are specially designed to identify long nucleotide sequence or find nearly identical sequences in different species.

Table 4.1. Programs Provided by Different Similarity Search Servers

Description	BLAST	FASTA	S&W SEARCH
DNA query vs. DNA	BLASTN,	FASTA3	SWN

database	MEGABLAST, discontiguous MEGABLAST		
Translated DNA query vs. protein database	BLASTX	FASTX3 FASTY3	SWX
Translated DNA query vs. translated DNA database	TBLASTX	N/A	TSWX
Protein query vs. protein database	BLASTP	FASTA3	SWP
Protein query vs. translated DNA database	TBLASTN	TFASTX3 TFASTY3	TSWN
Short pepetide query vs. protein database	N/A	FASTF3, FASTS3	N/A

4.3 Databases

Although obtaining as many hits as possible sounds attractive, it is very common that researchers are only interested in searching specific databases because of their special interests. Different sequence similarity search servers not only have many common databases, but also provide unique ones. Because of the large number of available databases, difference among the databases is one of the major factors that account for the complexity of the decision tree.

4.3.1 DNA Databases

Table 4.2 provides a partial list of DNA databases available at the three servers. Both BLAST and FASTA have vector database for searching vector contamination and

Table 4.2. DNA Databases Provided at the Sequence Similarity Search Servers

Databases	BLAST	FASTA	S&W SEARCH
Comprehensive	NR (Non-redundant)	EMALL	DDBJ ALL
GSS	Yes	Yes	Yes
HTG	Yes	Yes	Yes
STS	Yes	Yes	Yes
Patent	Yes	Yes	Yes
Update	Yes	Yes	Yes
Vector	Yes	Yes	No
Alu_repeat	Yes	No	No
Immunogenetic	No	Yes	No
Genome	Yes	Yes	No
WGS	Yes	Yes	No
EST	Yes	Yes	Yes
Species-Specific	Yes	Yes	Yes
Synthetic DNA	No	No	Yes
SNP	No	Yes	No

species-specific databases, while S&W SEARCH does not. FASTA also has single nucleotide polymorphism (SNP) database and databases specific for immuogenetic genes. BLAST is the only server supporting searches against alu_repeat database. BLAST and FASTA also have raw sequence databases, including genome database, whole genome shotgun (WGS) database, express sequence tag (EST) database, and databases derived from other high throughput sequencing projects such as bacterial artificial clone (BAC) and yeast artificial clone (YAC). BLAST and S&W SEARCH provide species specific EST databases, but with slightly different classification schemes.

4.3.2 Protein Databases

Table 4.3 provides a partial list of protein databases available at the three servers. The databases on the three servers are quite different. Except comprehensive, swiss-prot and pdb databases, all the other databases are only provided by one or two servers. For example, S&W SEARCH does not provide databases of patented sequence, but has several unique databases including PIR, PRF and DAD. BLAST is the only one with database of only most updated sequences. FASTA also provides UniRef databases which combine highly related protein sequences into a single record, which allows fast search with comprehensive database.

4.4 Server Specific Functions

Each server provides some unique functions for specific needs. For example, when conducting BLAST search, researchers are allowed to provide their own position

Table 4.3. Protein Databases Provided at the Sequence Similarity Search Servers

Database	BLAST	FASTA	S&W SEARCH
Comprehensive	Yes	Yes	Yes
Swiss-Prot	Yes	Yes	Yes
Patent	Yes	Yes	No
PDB	Yes	Yes	Yes
Update	Yes	No	No
PIR	No	No	Yes
PRF	No	No	Yes
DAD	No	No	Yes
UniRef	No	Yes	No
Prints	No	Yes	No
IPI	No	Yes	No

from position-specific iterative BLAST (PSI-BLAST). BLAST also allows search against databases satisfying certain query keywords. FASTA supports search of query sequence with specified range against database sequences with certain length.

4.5 Sequence Characteristics

For query sequences with different characteristics, such as length, sequence composition, and sources the sequence was obtained, parameter settings are essential for optimal search results. The most important parameters include scoring matrix, gap penalty, low-complexity filtering, word size (or ktup), and statistical significance. Right now, only BLAST provides default parameter settings for special search options based on the length of protein query sequence (Table 4.4). Similar settings can be used when translated DNA queries are searched against protein or translated DNA databases.

Table 4.4. Parameter Settings for Different Length of Protein Query Sequences in BLAST Search

Query Length (amino acid)	Scoring Matrix	Gap Penalty (start, extension)	Low Complexity Filter	E value	Word Size
< 35	PAM30	(9,1)	Off	1000	2
35 - 50	PAM70	(10,1)	On	10	3
50 - 85	BLOSUM80	(10,1)	On	10	3
> 85	BLSOUM62	(11,1)	On	10	3

4.6 Program Performance

S&W SEARCH implemented Smith-Waterman algorithm, which is the most rigorous algorithm and does not place any heuristic restrictions on the evolutionary model [B1]. Both BLAST and FASTA place additional restrictions on the alignments that they report in order to speed up their operation. The actual pattern of evolutionary changes between the query sequence and any homologues in the database can be incompatible with the heuristic restrictions imposed by either BLAST or FASTA.

Because of the mathematical rigor and lack of restrictions, the Smith-Waterman algorithm is more sensitive than either BLAST or FASTA [B2]. This additional sensitivity comes at the price of being a very much slower way to search a sequence database than either BLAST or FASTA [B2]. Sensitivity and selectivity of FASTA is between BLAST and Smith-Waterman-based programs [B2].

4.7 Summary

All the related issues in sequence similarity search listed in Table 4.1 - 4.4 were considered while the populated decision tree was implemented in this project, resulting in a decision tree with over 400 nodes. The knowledge on sequence similarity search is mainly obtained on the documentations or tutorials provided by the three sequence similarity search servers, as well as literatures discussing sequence similarity search [B2;B6]. The tutorials or documentations for BLAST, FASTA, and S&W Search programs can be found at http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/information3.html, http://www2.igh.cnrs.fr/help/fasta-help.html, http://www.ddbj.nig.ac.jp/search/help/swsearch-e_help.html, respectively. As the sequence similarity search algorithms being developed and servers continuously updated, it is necessary to expand or update our decision tree. User-interfaces are provided by the current project to allow users to extend the decision tree at the last level, or leaf nodes.

Although the knowledge contained in our decision tree likely represents the expert opinions of sequence similarity searches, some researchers might have different opinions. In such cases, functions are provided by our system to allow users to create their own decision tree. An almost empty decision tree is served as a template for users to expand based on their own preference. See Section 3.4.3 for more details on how this empty tree is designed and Section 6.2.6 for how to populating the empty tree.

CHAPTER FIVE

SOFTWARE QUALITY ASSURANCE

5.1 Introduction

Software quality assurance is a planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes, and procedures. Here, we only focus on using of software testing to evaluate the system.

5.2 Unit Test

Unit test is the initial step in software testing phase. It is to test the system's basic components regarding functionality, validation and compatibility. In S^4 expert system, these components are normally implemented in a single file (including Java, XML, Jess batch file or JSP) and tested individually before integration with other functions or system. The report of the unit test for S^4 is shown in Table 5.1. To facilitate illustration, tests are classified into categories. The test's goals as well as the files on which the test is conducted are listed. All the functions listed pass the test successfully.

Table 5.1. Report of S^4 Unit Test

Category	Test objective	Source Code	Result
XML file	XML validation based on corresponding DTD data structure definition.	S4DecisionTree.xml Help.xml	Pass
XML parser	Parse XML content by Java	S4DecisionTreeParser. java S4HelpParser.java	Pass
Java bean classes	Properties getter and setter function; Auto-decision node rule generation function.	csusb.s4.bean package	Pass
Jess expert system	1. Facts' construction from bean objects; 2. Process of decision tree traverse; 3. Process of decision tree update; 4. System control process.	S4JessExpert.java S4expert.CLP	Pass
Servlet	1. Proper display of page generated; 2. Links working as expected; 3. Error message display on error page.	S4ExpertDecisionServl et.java	Pass

Sequence Information	1. Sequence input preprocessing; 2. Sequence type determination.	S4InformationNode.java	Pass
XML update	Update XML content as expected.	S4DecisionTreeExpander.java	Pass
S^4 decision learner	Control process for a sub-tree to be added in depth-first order.	S4ExpertLearner.java	Pass
JSP page	1. Page content display correctly; 2. Links working as expected; 3. Error handling;	JSP pages involved in sequence input, option selection, suggestion pages as well as decision tree extension pages.	Pass
Remote Server data integration	Data integration with remote similarity search services, if support by the remote server.	S4RemoteSever_blast.jsp	pass

5.3 Integration Test

Most of the functionalities in the system are provided by integrating some of the individual components. The function-related components are integrated to some extent and tested based on different functionalities. The test report is shown in Table 5.2. The integrated components with the tested functions are listed. All the integration tests were passed as desired.

Table 5.2. Report of S^4 Integration Test

Test components	Test objective	Report
Decision tree XML parser + bean classes	Parse XML document to bean objects	Pass
Expert System + bean classes	Expert system knowledge facts construction using bean objects as data. Expert system knowledge rules construction after adding rules for auto-decision node.	Pass
Expert System + servlet + IOExchange	Message passing by IOExchange object	Pass
Expert System + servlet	Data exchange by Java object.	Pass
XML update + bean classes	Update XML based on information provided by bean objects.	Pass
Decision tree extension + bean classes	Decision tree extension by input information provided by bean objects.	Pass
Decision tree extension + XML update + bean objects	XML content update during decision tree extension via bean objects.	Pass
Decision tree extension + expert system + bean objects	Expert system knowledge extension during decision tree extension via bean objects	Pass

JSP pages + bean classes	Bean class usage for JSP form content input to or retrieval from bean object.	Pass
JSP pages + servlet	Form information and request information sent to servlet after invoked.	Pass
Help page + help XML parser	Proper display help page content based on information parsed from help XML page.	Pass
Remote services + S^4 system	Automatically setting remote services parameters for suggested in S^4 expert system	Pass (BLAST search only)

5.4 System Test

The whole system is created by integrating all the components to fulfill all the functionalities of S^4. These functions include:

1. Submit sequence query online.
2. Select option based on their requirement and use the given suggestions to do sequence similarity search by suggested program.
3. Improve expert system knowledge by extending decision tree.
4. Repeat option selection using the same sequence input.

5. Exit the system.

All the functions are tested and passed successfully, as shown in Table 5.3.

Table 5.3. Report of S^4 System Test

Function	Test objective	Result
Sequence input	1. Correct sequence pre-processing process; 2. Error handling.	Pass
Expert System Knowledge construction	Knowledge constructed from right source (XML or Jess-stored text file).	Pass
Option selection	1. Proper knowledge display from expert system; 2. Reliable communication between user and expert system; 3. Auto-decision node processing correctly.	Pass
Suggestion display	Suggested settings given as expected.	Pass
Remote server connection	1. Connected to remote server as expected. 2. If remote server is BLAST, suggested settings are set on remote search page	Pass
Option selection for same sequence input.	1. Redisplay the options correctly. 2. Suggested settings given changed as different options selected.	Pass

Decision tree extension	1. Correct form input. 2. Extensive input error handling. 3. Depth-first order node information input. 4. Decision tree update in XML file. 5. Current Jess expert system knowledge update upon extension. 6. Immediately take effective on current expert system.	Pass
System exit	Expert system proper termination upon user's system exit.	Pass

5.5 Test Sampling

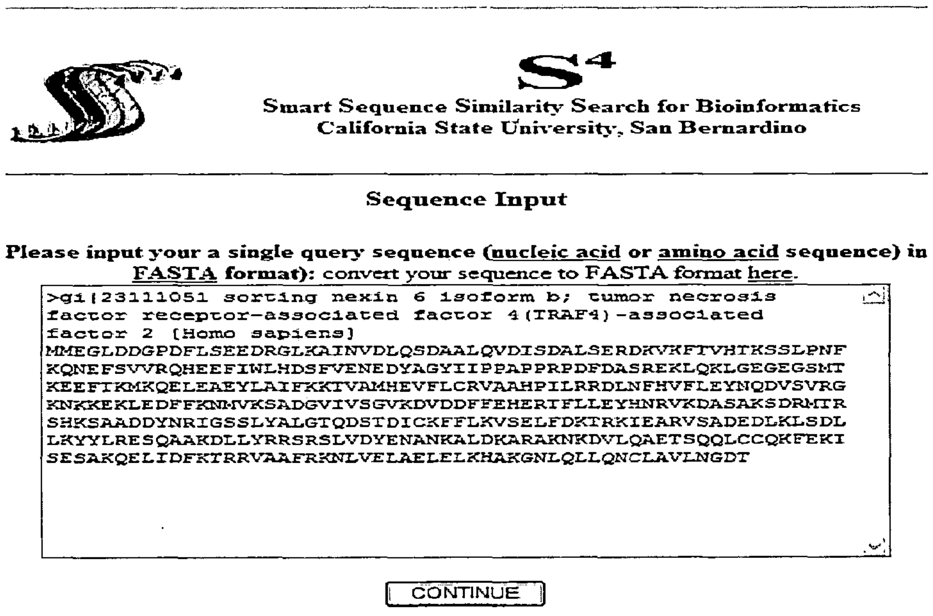
In this section, samples of test case are listed individually.

5.5.1 Search with Sequence Query

1. Submit the query sequence (Figure 5.1)

2. The system determines the query sequence is a protein, so the amino acid sequence query goal page is displayed to the user (Figure 5.2).

3. After selecting options to find similar protein sequence with comprehensive database and with specific species, the suggestion page displayed the parameter settings as expected (Figure 5.3).


S4

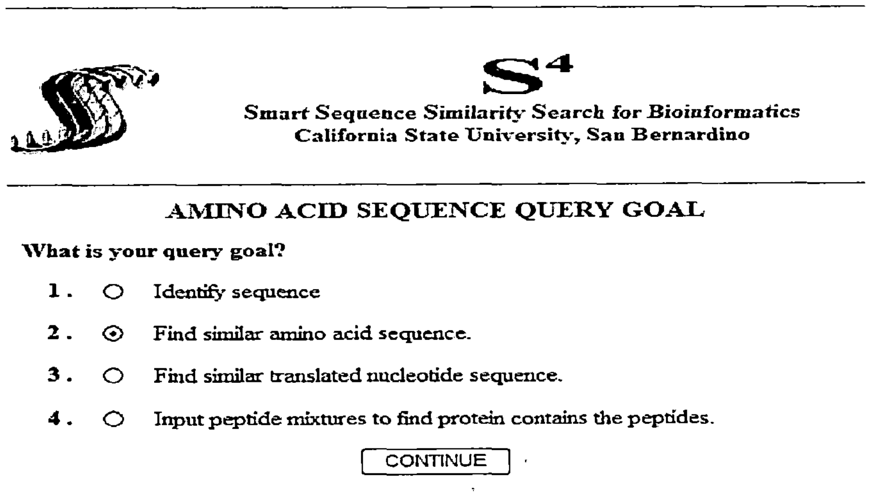
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Sequence Input

Please input your a single query sequence (nucleic acid or amino acid sequence) in FASTA format): convert your sequence to FASTA format here.

```
>gi|23111051 sorting nexin 6 isoform b; tumor necrosis
factor receptor-associated factor 4(TRAF4)-associated
factor 2 [Homo sapiens]
MMEGLDDGPDFLSEEDRGLKAINVDLQSDAALQVDISDALSERDKVKFTVHTKSSLPNF
KQNEFSVVRQHEEFIWLHDSFVENEDYAGYIIPPAPPRPDFDASREKLQKLGEGEGSMT
KEEFTKMKQELEAEYLAIFKKTVAMHEVFLCRVAAHPILRRDLNFHVFLEYNQDVSVRG
KNKKEKLEDFFKNMVKSADGVIVSGVKDVDDFFEHERTFLLEYHNRVKDASAKSDRMTR
SHKSAADDYNRIGSSLYALGTQDSTDICKFFLKVSELFDKTRKIEARVSADEDLKLSDL
LKYYLRESQAAKDLLYRRSRSLVDYENANKALDKARAKNKDVLQAETSQQLCCQKFEKI
SESAKQELIDFKTRRVAAFRKNLVELAELELKHAKGNLQLLQNCLAVLNGDT
```

Figure 5.1. Submit a Protein Sequence


S4

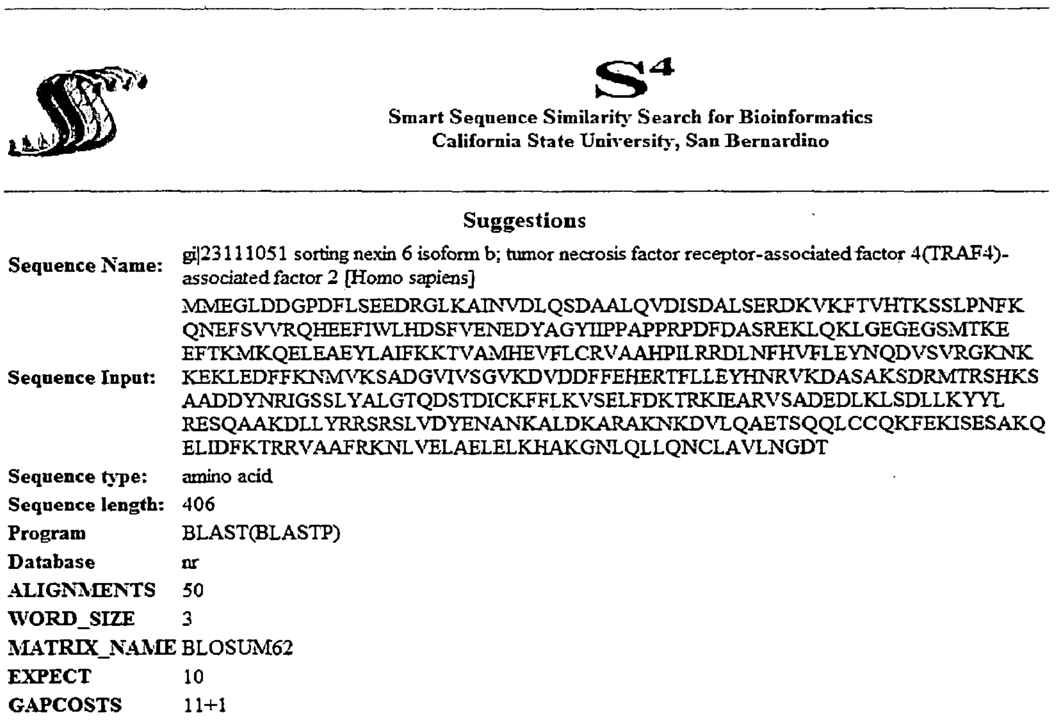
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

AMINO ACID SEQUENCE QUERY GOAL

What is your query goal?

1. ○ Identify sequence
2. ⦿ Find similar amino acid sequence.
3. ○ Find similar translated nucleotide sequence.
4. ○ Input peptide mixtures to find protein contains the peptides.

Figure 5.2. Amino Acid Sequence Query Goal Selection Page



S4

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

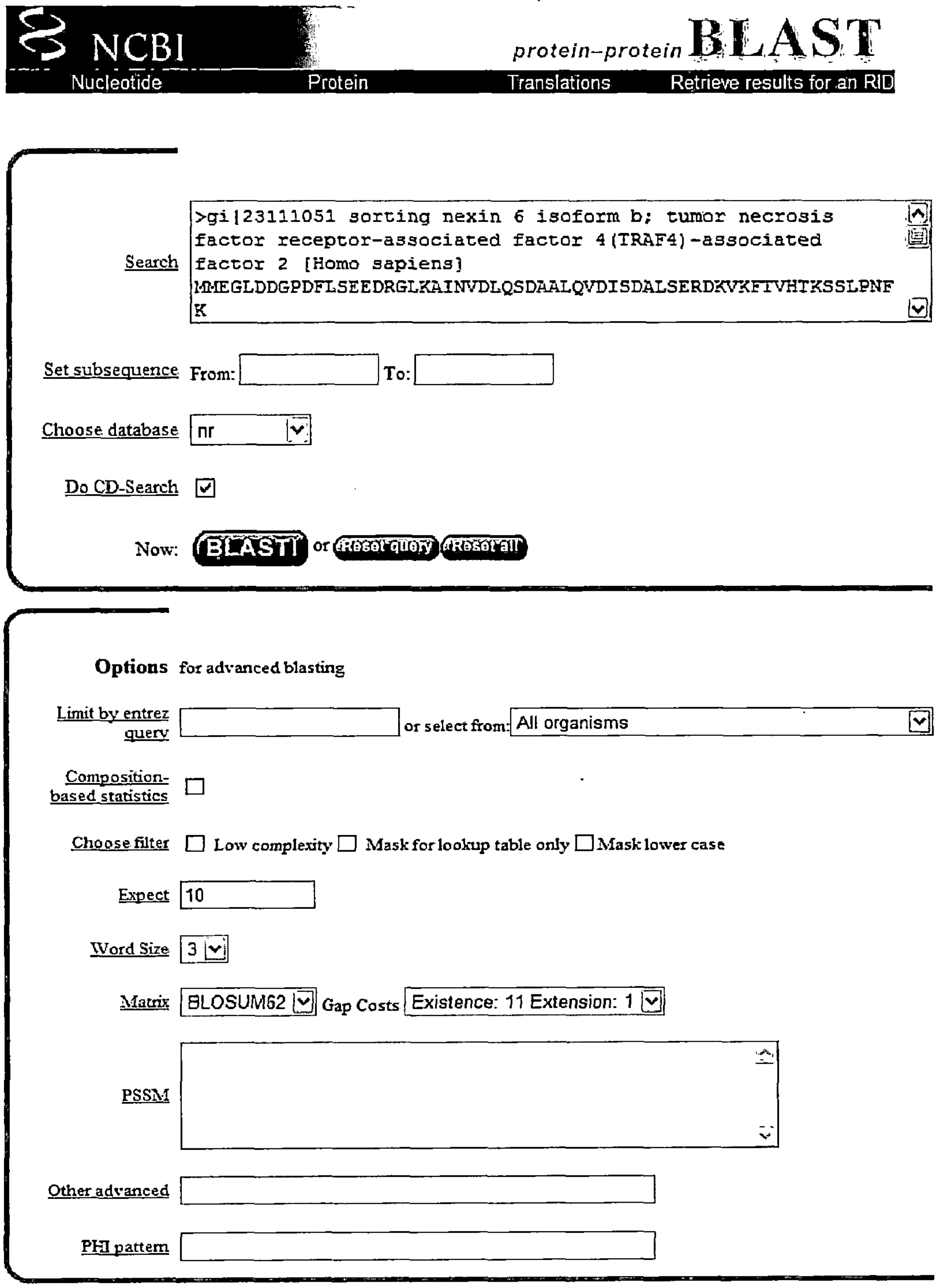
Suggestions

Sequence Name:	gi|23111051 sorting nexin 6 isoform b; tumor necrosis factor receptor-associated factor 4(TRAF4)-associated factor 2 [Homo sapiens]
Sequence Input:	MMEGLDDGPDFLSEEDRGLKAINVDLQSDAALQVDISDALSERDKVKFTVHTKSSLPNFK QNEFSVVRQHEEFIWLHDSFVENEDYAGYIIPPAPPRPDFDASREKLQKLGEGEGSMTKE EFTKMKQELEAEYLAIFKKTVAMHEVFLCRVAAHPILRRDLNFHVFLEYNQDVSVRGKNK KEKLEDFFKNMVKSADGVIVSGVKDVDDFFEHERTFLLEYHNRVKDASAKSDRMTRSHKS AADDYNRIGSSLYALGTQDSTDICKFFLKVSELFDKTRKIEARVSADEDLKLSDLLKYYL RESQAAKDLLYRRSRSLVDYENANKALDKARAKNKDVLQAETSQQLCCQKFEKISESAKQ ELIDFKTRRVAAFRKNLVELAELELKHAKGNLQLLQNCLAVLNGDT
Sequence type:	amino acid
Sequence length:	406
Program	BLAST(BLASTP)
Database	nr
ALIGNMENTS	50
WORD_SIZE	3
MATRIX_NAME	BLOSUM62
EXPECT	10
GAPCOSTS	11+1

Figure 5.3. Suggestion Page for Searching Similar Proteins from a Specific Species

4. Conduct search with suggested settings in a new window. Since the suggested program is BLASTP, our system supports automatic settings on the remote BLASTP search page (Figure 5.4).

5. If the search goal is to find similar translated DNA sequences from bacteria or archea, but limited to the most recent updated database, we choose the function in the above suggestion page to try again. After the desired options are selected, the new suggested settings are displayed as shown in Figure 5.5.



Search: >gi|23111051 sorting nexin 6 isoform b; tumor necrosis factor receptor-associated factor 4(TRAF4)-associated factor 2 [Homo sapiens]
MMEGLDDGPDFLSEEDRGLKAINVDLQSDAALQVDISDALSERDKVKFTVHTKSSLPNFK

Set subsequence From: To:

Choose database: nr

Do CD-Search: ✓

Now: BLAST! or Reset query Reset all

Options for advanced blasting

Limit by entrez query: or select from: All organisms

Composition-based statistics: ☐

Choose filter: ☐ Low complexity ☐ Mask for lookup table only ☐ Mask lower case

Expect: 10

Word Size: 3

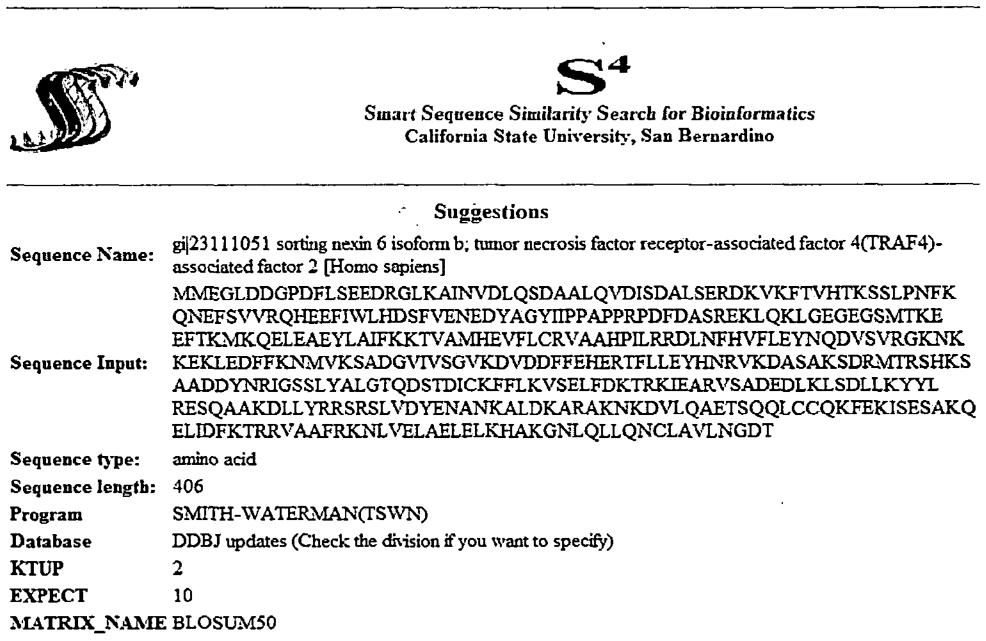
Matrix: BLOSUM62 Gap Costs: Existence: 11 Extension: 1

PSSM

Other advanced

PHI pattern

Figure 5.4. BLASTP Search Page with the Suggested Settings


S⁴

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

Suggestions

Sequence Name:	gi|23111051 sorting nexin 6 isoform b; tumor necrosis factor receptor-associated factor 4(TRAF4)-associated factor 2 [Homo sapiens]
Sequence Input:	MMEGLDDGPDFLSEEDRGLKAINVDLQSDAALQVDISDALSERDKVKFTVHTKSSLPNFK QNEFSVVRQHEEFIWLHDSFVENEDYAGYIIPPAPPRPDFDASREKLQKLGEGEGSMTKE EFTKMKQELEAEYLAIFKKTVAMHEVFLCRVAAHPILRRDLNFHVFLEYNQDVSVRGKNK KEKLEDFFKNMVKSADGVIVSGVKDVDDFFEHERTFLLEYHNRVKDASAKSDRMTRSHKS AADDYNRIGSSLYALGTQDSTDICKFFLKVSELFDKTRKIEARVSADEDLKLSDLLKYYL RESQAAKDLLYRRSRSLVDYENANKALDKARAKNKDVLQAETSQQLCCQKFEKISESAKQ ELIDFKTRRVAAFRKNLVELAELELKHAKGNLQLLQNCLAVLNGDT
Sequence type:	amino acid
Sequence length:	406
Program	SMITH-WATERMAN(TSWN)
Database	DDBJ updates (Check the division if you want to specify)
KTUP	2
EXPECT	10
MATRIX_NAME	BLOSUM50

Figure 5.5. Suggestion Page for Searching Similar Translated DNAs from Most Updated Bacteria or Archea Databases

6. Improve the expert system knowledge by updating the decision tree. The example subtree to be extended to the current leaf node is shown in Figure 5.6.

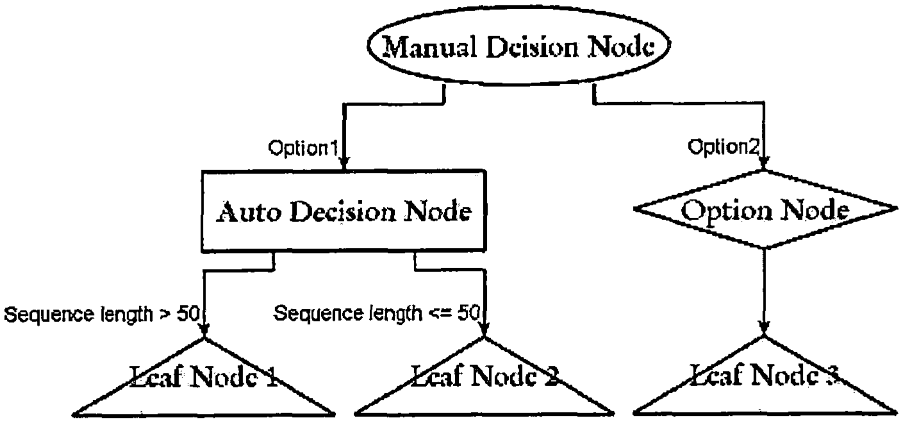
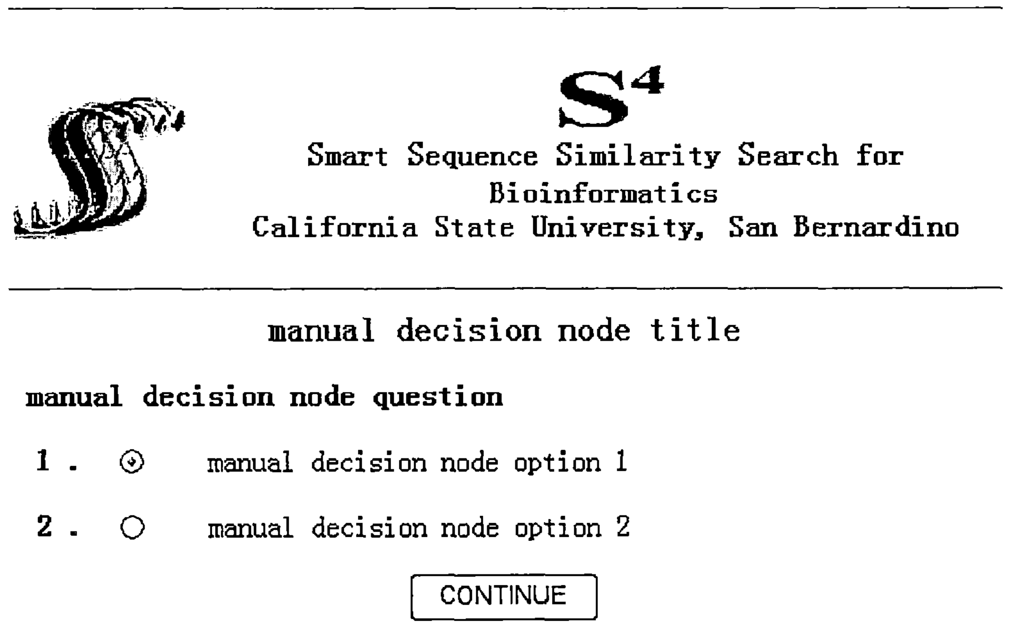


Figure 5.6. Example Subtree to be Extended

After inputting all the nodes in the above subtree in depth-first order and updating the decision permanently, we tested the system with the same query sequence and same search options as in step 5. A new option selection page reflecting the updated decision tree is displayed, see Figure 5.7.



S4

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

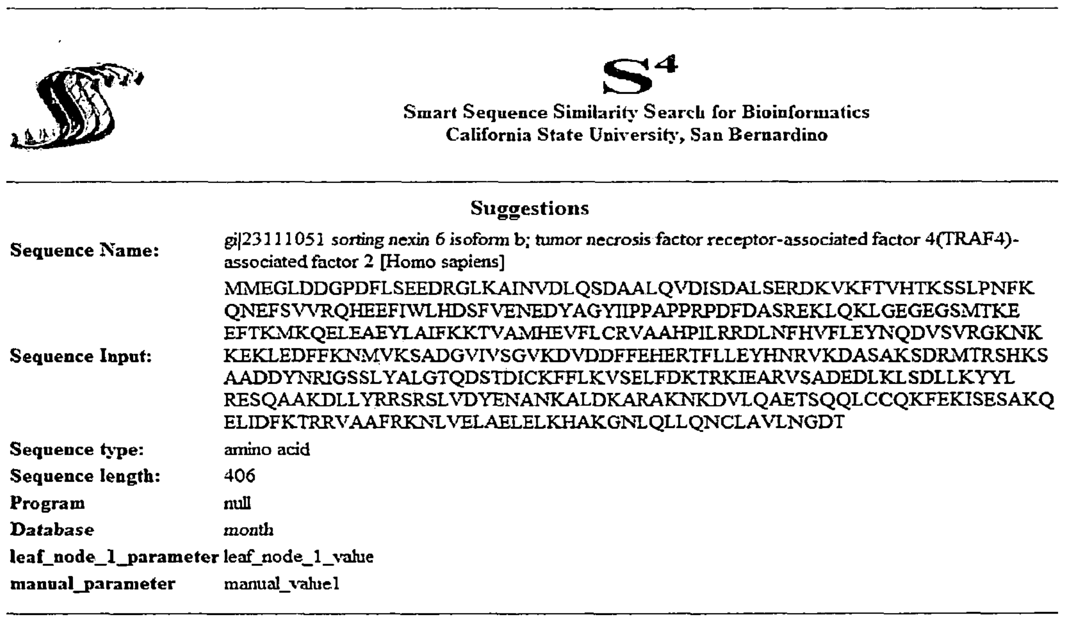
manual decision node title

manual decision node question

1. [✓ ◉] manual decision node option 1
2. ○ manual decision node option 2

Figure 5.7. New Option Selection Page

After the option 1 is selected, the suggestion page shows the expected settings pointing to leaf_node_1 (Figure 5.8), resulting from the automatic decision of option 1 (sequence length is more than 50) of the auto_decision_node (Figure 5.6).



S⁴

Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

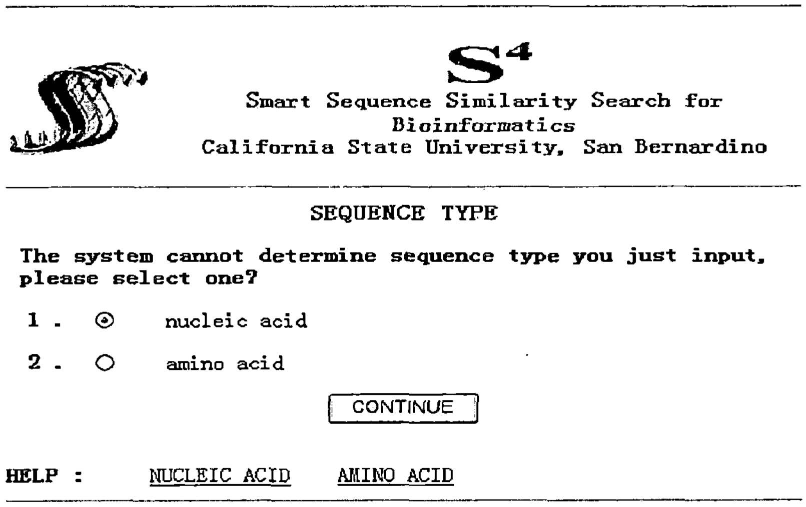
Suggestions

Sequence Name:	gi|23111051 *sorting nexin 6 isoform* b; *tumor necrosis factor receptor-associated factor* 4(TRAF4)-associated factor 2 [Homo sapiens]
Sequence Input:	MMEGLDDGPDFLSEEDRGLKAINVDLQSDAALQVDISDALSERDKVKFTVHTKSSLPNFK QNEFSVVRQHEEFIWLHDSFVENEDYAGYIIPPAPPRPDFDASREKLQKLGEGEGSMTKE EFTKMKQELEAEYLAIFKKTVAMHEVFLCRVAAHPILRRDLNFHVFLEYNQDVSVRGKNK KEKLEDFFKNMVKSADGVIVSGVKDVDDFFEHERTFLLEYHNRVKDASAKSDRMTRSHKS AADDYNRIGSSLYALGTQDSTDICKFFLKVSELFDKTRKIEARVSADEDLKLSDLLKYYL RESQAAKDLLYRRSRSLVDYENANKALDKARAKNKDVLQAETSQQLCCQKFEKISESAKQ ELIDFKTRRVAAFRKNLVELAELELKHAKGNLQLLQNCLAVLNGDT
Sequence type:	amino acid
Sequence length:	406
Program	null
Database	*month*
leaf_node_1_parameter	leaf_node_1_value
manual_parameter	manual_value1

Figure 5.8. Suggestion Page after Updating the Decision Tree

5.5.2 Auto Decision Node Handling

When the system can not determine which option is satisfied based on the rules of an auto decision node, the options are displayed to the user for selection to guarantee the proper function of the system. For example, when the input query sequence can not be determined whether it is a DNA or protein, a page will be prompted to the user to select the type of the sequence (Figure 3.9). This might happen when there are special characters in the query sequence. The content of the page is based on the information contained in the auto decision node.



S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

SEQUENCE TYPE

The system cannot determine sequence type you just input, please select one?

1 . [✓ nucleic acid]

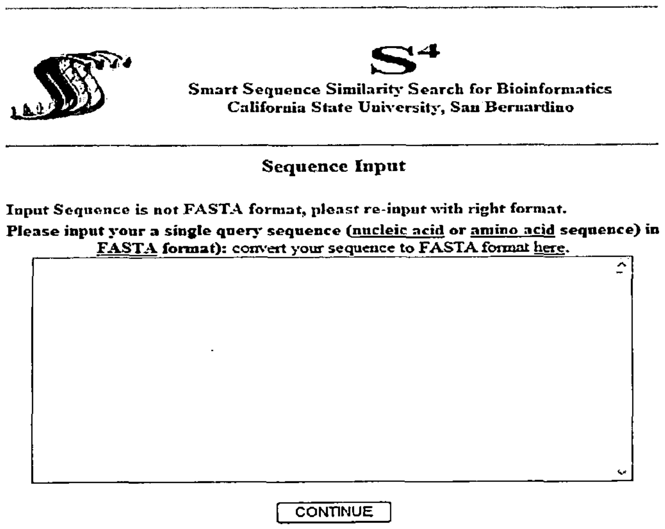
2 . amino acid

HELP : NUCLEIC ACID AMINO ACID

Figure 5.9. Sequence Type Selection Page

5.5.3 Error Handling

5.5.3.1 Sequence Input Page Error Handling. When the input sequence is not in Fasta format, an error page is displayed (Figure 5.10).



S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

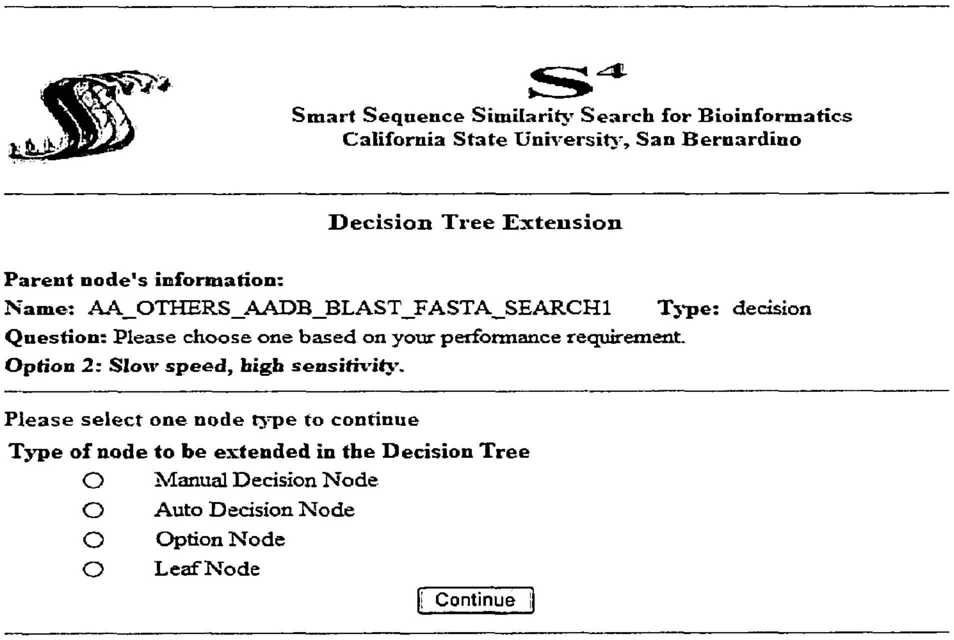
Sequence Input

Input Sequence is not FASTA format, pleast re-input with right format.
Please input your a single query sequence (nucleic acid or amino acid sequence) in FASTA format): convert your sequence to FASTA format here.

Figure 5.10. Input Sequence Page Error Handling

5.5.3.2 Error Handling in Decision Tree Extension.

5.5.3.2.1 Node Type Selection. When none of the node types is selected, an error page will be displayed (Figure 5.11).


S4
Smart Sequence Similarity Search for Bioinformatics
California State University, San Bernardino

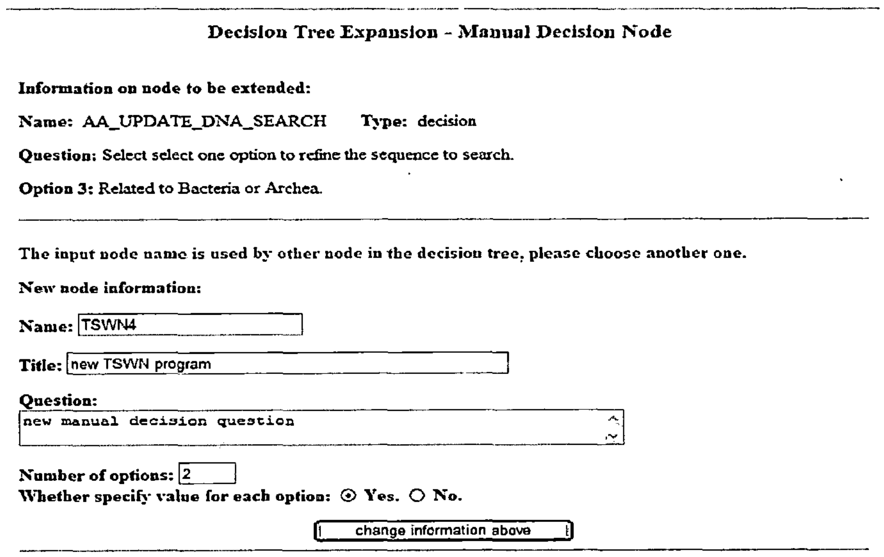
Decision Tree Extension

Parent node's information:
Name: AA_OTHERS_AADB_BLAST_FASTA_SEARCH1 **Type:** decision
Question: Please choose one based on your performance requirement.
Option 2: Slow speed, high sensitivity.

Please select one node type to continue
Type of node to be extended in the Decision Tree

Figure 5.11. Error Handling on Node Type Selection

5.5.3.2.2 New Node Information Input. When the information is submitted, the server checks whether all the required information is provided, the validity of the provided information. Example error messages are listed in Figure 5.12 and 5.13.


Decision Tree Expansion - Manual Decision Node

Information on node to be extended:

Name: AA_UPDATE_DNA_SEARCH **Type:** decision

Question: Select select one option to refine the sequence to search.

Option 3: Related to Bacteria or Archea.

The input node name is used by other node in the decision tree, please choose another one.

New node information:

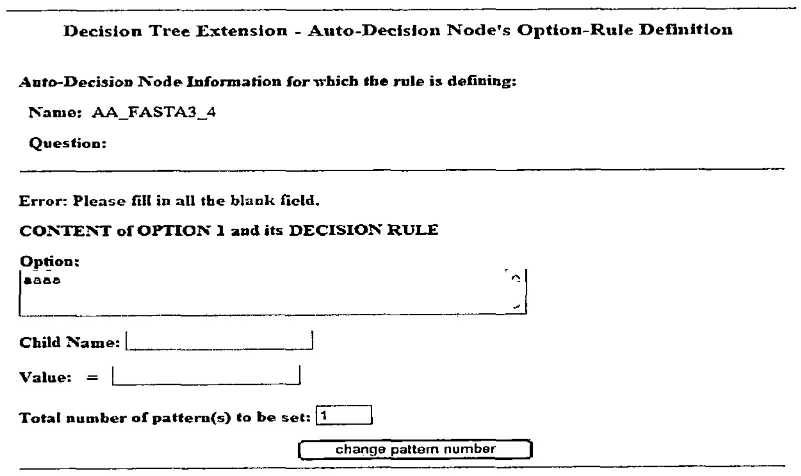
Name: TSWN4

Title: new TSWN program

Question:
new manual decision question

Number of options: 2
Whether specify value for each option: [✓ Yes.] No.

Figure 5.12. Handling the Node Name Validity


Decision Tree Extension - Auto-Decision Node's Option-Rule Definition

Auto-Decision Node Information for which the rule is defining:

Name: AA_FASTA3_4

Question:

Error: Please fill in all the blank field.

CONTENT of OPTION 1 and its DECISION RULE

Option:
aaaa

Child Name:

Value: =

Total number of pattern(s) to be set: 1

Figure 5.13. Handling the Required Fields

CHAPTER SIX

MAINTENANCE AND USERS MANUAL

6.1 Maintenance Manual

This section contains all the structures and directories of files including source code, object codes as well as documentations. It also contains instructions on how to build and reinstall the program.

6.1.1 Directory Organization

Under the S^4 directory, there are twenty one JSP pages and five subdirectories: doc, jess_code, src, WEB_INF, and xml.

1. /doc: This directory contains documentation for the S^4 system.
2. /jess_code: This directory contains all the text files used in Jess expert system.
    a) S4expert.CLP: Batch file containing rules related to Jess expert system.
    b) initial/Auto_decision_rules.CLP and expert/Auto_decision_rules.CLP: Text files containing rules related to option selection of auto decision nodes for expert users and common users respectively.

c) Initial/Decision_making_facts.CLP and expert/Decision_making_facts.CLP: Text files containing facts representing knowledge in decision tree for expert users and common users respectively.

3. /src: This directory contains the JAVA source code that generate the servlets, beans, and other Java classes that are unique to the S^4 system. The files are organized into the same subdirectories as those in /WEB_INF to reflect the package hierarchy.

4. /WEB_INF: This directory contains necessary executable files for server running. There are two subdirectories, classes and lib.

   a) /WEB_INF/classes: This subdirectory contains all the Java class files (and associated resources) required for S^4 system. All the java class files are located in /WEB_INF/classes/csus/s4/ with four subdirectories: bean, system, utility, and xml.

      i. Bean: Classes to present nodes in decision tree.

         1) S4GeneralNode: Class for common properties of all types of nodes.

2) S4InternalNode: Class for common properties of an interior node.

3) S4LeafNNode: Class for properties of a leaf node.

4) S4OptionNode: Class for properties of an option node.

5) S4DecisionNode: Class for common properties of manual and auto decision node.

6) S4AutoDecisionNode: Class for properties auto decision node.

7) S4DecisionRule: Class for rule definition for an option of an auto-decision node.

8) S4DesisicionRulePattern: Class for pattern definition for an auto-decision rule.

ii. System:

1) S4JessExpert: Class for Jess expert system thread.

2) S4JessExpert$EventHandler: Inner class in S4JessExpert to communicate with Servlet.

3) S4ExpertDecisionServlet: Servlet for displaying information from jess expert system by communicating with Jess.

4) S4ExpertLearner: Class for expert system knowledge learning process.

5) IOExhange: Class for communication between S4JessExpert and Servlet.

6) S4InformationNode: Class for storing useful information in decision making.

7) S4Exception: Class for exception handling.

iii. Xml:

1) S4DecisionTreesParser: Class for parsing decision tree in XML to bean objects.

2) S4DecisionTreesExpander: Class for extending decision tree in XML from a specified leaf node.

3) S4HelpParser: Class for parsing help content in XML.

iv. Utility:

1) ArrayUtility: Class for static functions related to arrays.

b) /WEB_INF/lib: This directory contains JAR files that contain Java class files (and associated resources) required for S^4 system. There is only one JAR file called jess.jar which provides necessary libraries to support the expert system.

5. /xml: This directory contains the information used in S^4 system in XML format and their corresponding DTD file.

   a) initial/S4Decisiontrees.DTD and expert/ S4Decisiontrees.DTD: Files to define data structure of S^4 decision tree for expert and common users, respectively.

   b) initial/S4Decisiontree.xml and expert/ S4Decisiontree.xml: Files containing knowledge in the form of S^4 decision tree structure defined in S4DecisionTrees.DTD file for expert and common users, respectively.

   c) S4Help.DTD: File to define data structure for representing help information.

   d) S4Help.xml: File containing help information in the format defined in S4Help.DTD.

6. JSP pages: The JSP pages for user interfaces can be classified into several categories.

a) SequenceInput.jsp: Page for sequence input.

b) S4OutputResult.jsp: Page for display suggestions and further options.

c) Pages for nodes' information input/edit pages and confirmation pages:

d) Pages for controlling decision tree extension:

   i. S4DecisionTreeExtend_start.jsp: Page for start extension.

   ii. S4DecisionTreeExtendcontrol.jsp: Page for controlling the decision tree extension process.

   iii. S4DecisionTreeExtend_control_submit.jsp: Page for storing newly-input node.

   iv. S4DecisionTreeConfirm.jsp: Pages to confirm the decision tree extension.

   v. S4DecisionTreeFinish.jsp: Pages to display after decision tree extension successfully.

6.1.2 Re-Compile the S^4 System Source Code

The source code can be re-compiled by "javac" command whenever there are changes in any of the source files. First of all, the class path should be specified to allow "javac" to find third-party and user-defined classes -- that is, classes that are not Java extensions or part of the Java platform. "-d" options should be specified to set the destination directory for class files. Within the destination directory, "javac" puts the class file in a subdirectory under reflecting the package name, creating directories as needed. For example, S4ExpertDecisionServlet.java which specifies the package name as csusb.s4.system has been edited. To re-compile it, go to the /src directory, and type the command "javac -classpath jess.jar -d ../WEB_INF S4ExpertDecisionServlet.java", which will compile the source file and put S4ExpertDecisionServlet.class into the directory /WEB_INF/csusb/s4/system.

6.1.3 Installing the S^4 System on Tomcat Server

In order to be executed, S^4 system must be deployed on a servlet container. We will describe using Tomcat 4 to provide the execution environment. Deploying on other servlet containers will be specific to each container. Please consult the user manual of the servlet container of your choice. S^4 system can be deployed in Tomcat by the following steps:

1. Create a subdirectory in directory $CATALINA_HOME/webapps/ and name it as S4;
2. Copy the unpacked directory hierarchy of S^4 system into above subdirectory;

3. Add a <Context> entry for the S^4 system in the configuration file, $CATALINA_HOME/conf/server.xml.
4. Restart Tomcat after installing the S^4 system application.

6.2 Users Manual

This section contains the user manual to guide the user to use the service properly.

6.2.1 Input a Sequence

You can copy and paste their query sequence into the box provided in the sequence input page. The sequence can be either DNA or protein, but must be in Fasta format. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. An example sequence in FASTA format is:

>gi|23111051 sorting nexin 6 isoform b; tumor necrosis factor receptor-associated factor 4(TRAF4)-associated factor 2 [Homo sapiens]

```
MMEGLDDGPDFLSEEDRGLKAINVDLQSDAALQVDISDALSERDKVKFTVHTK
SSLPNFKQNEFSVVRQHEEFIWLHDSFVENEDYAGYIIPPAPPRPDFDASREKLQKLG
EGEGSMTKEEFTKMKQELEAEYLAIFKKTVAMHEVFLCRVAAHPILRRDLNFHVFLEY
NQDVSVRGKNKKEKLEDFFKNMVKSADGVIVSGVKDVDDFFEHERTFLLEYHNRVKDA
```

SAKSDRMTRSHKSAADDYNRIGSSLYALGTQDSTDICKFFLKVSELFDKTRKIEARVS
ADEDLKLSDLLKYYLRESQAAKDLLYRRSRSLVDYENANKALDKARAKNKDVLQAETS
QQLCCQKFEKISESAKQELIDFKTRRVAAFRKNLVELAELELKHAKGNLQLLQNCLAV
LNGDT

6.2.2 Search Option Selection

You will be prompted with a series of questions and asked to select one option for each question. The options were built based on expert knowledge of sequence similarity search. Press "continue" button to go to the next page after selecting the desired option. The suggested settings will be displayed when the system thinks there is no more questions need to ask.

6.2.3 Search against the Remote Server with the Suggested Settings

To conduct the search based on the suggested settings, you simply click the button "Open another window to do search by suggested server" on the suggestion page. This will open a new window which links you to the suggested sequence similarity search server with the suggested parameter settings. Note that only BLAST server supports this function. For EBI FASTA and DDBJ SSEARCH, you need to set the parameters on their page with the ones suggested by our system.

6.2.4 Change the Search Options

If you would like to do another search, but with different search options with the same input sequence, simply click the button "Try again with the same sequence" on the suggestion page. This will allow you to go over the series of questions again regarding to your search options.

6.2.5 Exit the System

When you are done with the system, click the button "Exit the system now" on the suggestion page. You will be linked to another page and have the option to either close the browser window or do another search with a different input sequence.

6.2.6 Improve the Expert Knowledge of S^4 System

You can improve the performance of S^4 system to include your expert knowledge of sequence similarity search. This function is mainly provided to the advanced users who are considered as experts in sequence similarity search.

To be able to efficiently improve the expert knowledge, you must understand some behind-of-the-scenes. S^4 system contains the knowledge on Sequence Similarity Search in the form of facts and rules. This knowledge is organized and presented in a decision tree in S^4 system. A decision tree possesses (typically) a single root, a set of branches from that root, interior nodes that also have branches and leaf nodes, which present the classification information. Each node within the tree represents a decision point that determines which subsequent branch is followed until a leaf node is reached. The leaf nodes in the tree represent all the possible solutions that can be derived from the tree, which is also called answer node. Thus, a defined decision tree provides a user paradigm for solving certain type of classification problems, and finally providing the answer to a problem from a predetermined set of possible answers. The decision tree can be easily extended by deleting a leaf node and replaced it with interior nodes that each has open path ended with a newly defined leaf node.

There are three types of decision nodes, including Manual Decision Node (ellipse), Auto Decision Node (rectangle), and Option Node (diamond). The leaf nodes are in triangle.

As other decision trees, S^4 decision tree consists of nodes and branches (Figure 6.1). The leaf nodes (triangle) represent all the possible suggestions that the system can provide. All other nodes in the tree are referred to as decision nodes. To better represent the knowledge included in the S^4 decision tree, three types of decision nodes are constructed: manual decision node, auto decision node and option node.

1. Manual Decision Node (ellipse in Figure 6.1): The location of decision tree will move from this node to

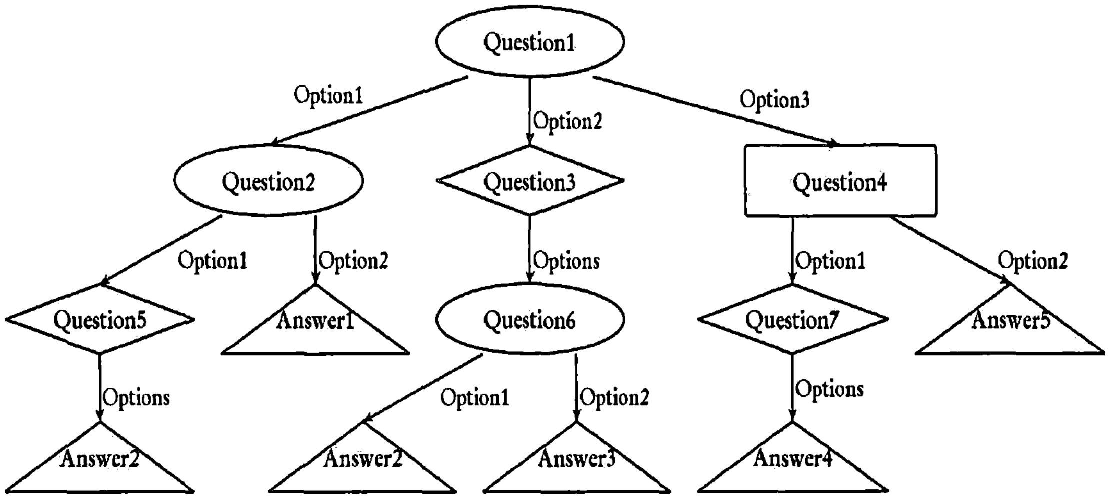


Figure 6.1. Nodes in the Decision Tree

corresponding child node based on the branch the user selects.

2. Auto Decision Node (rectangle): The location of decision will move from this node to corresponding child node based on the branch automatically selected by the system using predefined branch-specific criterion.

3. Option Node (diamond): No matter which branch the user selects, there is only one child node that the decision tree will move to. It is responsible for setting

the value for a specified parameter based on the user selection.

You can extend any type of nodes as you need until the leaf node is reached. Press "I want to extend the decision tree for better result" to go to the decision tree extension page. A confirmation page will be displayed to you before saving the changes permanently. You can either exit the system at this point, or test the newly updated system to do the search with the same sequence by clicking "Try again with the same sequence".

<u>6.2.6.1 Extend the Manual Decision Node</u>. Following these steps:

1. Select "Manual decision node" in the decision tree extension page;
2. Fill in the name, title and question associated with the node;
3. Put in any integer number in the box to the right of "Number of options". These options will be displayed to the user to choose. There can be multiple options associated with a manual decision node;
4. For above options, you should specify whether a search parameter will be known when any of the

options is chosen. If so, you should choose "Yes" button on the right hand side of "Whether a parameter should be set". Otherwise, choose "No";

5. Press "continue" to go to the next page;
6. One or more lines of boxes appear at the bottom of the page based on the number of options you input. Each line of boxes is the settings you should provide for each of the options associated with the manual decision node;
7. You should now input the option's content, its child node (where to go if the option is chosen), parameter name and values if you have chosen to specify parameters in step 3;
8. A confirmation page will be displayed to you showing all the settings you just inputted. Click "Go ahead and save the node" if the settings are correct. Click "GO back to the page to update the content" if you want to change the settings;
9. After you save the settings from step 8, you will be asked to set all the child nodes of the current node one by one. If the child node is again a manual decision node, go back to step 1. If the child node is an auto decision node, go to

Section 6.2.6.2. If the child node is an option node, go to Section 6.2.6.3. If the child node is a leaf node, go to Section 6.2.6.4.

6.2.6.2 Extend the Auto Decision Node. Following these steps:

1. Select "Auto decision node" in the decision tree extension page;
2. Fill in the name, title and question associated with the node;
3. You should specify whether a search parameter will be known when any of the options is chosen. If so, you should choose "Yes" button on the right hand side of "Whether a parameter should be set". Otherwise, choose "No". Click "Continue" to go to next page;
4. If you select to specify a parameter in the previous screen, you are asked to put in the parameter name. Click "Add new option" to start adding options;
5. Fill in the option content and name for the child node;

6. In the same page, fill in the number of criterion the system should check so that whether the option should be chosen. For example, one criterion can be that the length of the query sequence must be greater than 1000;
7. You can add more options by click "Add a new option" button;
8. If you have selected to specify a parameter for the options, now you should input a value for the parameter for each option;
9. You can modify or delete the current option by clicking buttons besides the option input form;
10. After all the options have been added, click "save node" button to save the node;
11. A confirmation page will be displayed to you showing all the settings you just inputted. Click "Confirm and Save the node" if the settings are correct. Click "Modify the node" if you want to change the settings;
12. After you save the settings from step 11, you will be asked to set all the child nodes of the current node one by one. If the child node is a manual decision node, go to Section 6.2.6.1. If

the child node is an auto decision node, go back to step 1. If the child node is an option node, go to Section 6.2.6.3. If the child node is a leaf node, go to Section 6.2.6.4.

6.2.6.3 Extend the Option Node. Following these steps:

1. Select "Option node" in the decision tree extension page;
2. Fill in the name, title and question associated with the node;
3. Put in any integer number in the box to the right of "Number of options". These options will be displayed to the user to choose. There can be multiple options associated with an option node; Press "Continue" to the next screen;
4. Based on the number of options you input, one or multiple lines of boxes are displayed to you. You need to fill in each line of boxes for each option;
5. Since there is always one parameter associated with the option node, you are asked to input its name and its value associated with each option;

6. There is exactly one child node associated with each option node, so you are asked to input the child node name; Press "Continue";

7. A confirmation page will be displayed to you showing all the settings you just inputted. Click "Confirm and Save the node" if the settings are correct. Click "Modify the node" if you want to change the settings;

8. After you save the settings from step 7, you will be asked to set all the child nodes of current node. If the child node is a manual decision node, go to Section 6.2.6.1. If the child node is an auto decision node, go to Section 6.2.6.2. If the child node is an option node, go to Section 6.2.6.3. If the child node is a leaf node, go to Section 6.2.6.4.

6.2.6.4 Extend the Leaf Node. Following these steps:

1. Select "Option node" in the decision tree extension page;

2. Put in number of parameters should be set for this leaf node; Click "Continue" to go to the next screen;

3. You are asked to provide the parameter names and values;
4. A confirmation page will be displayed to you showing all the settings you just inputted. Click "Go ahead and save the node" if the settings are correct. Click "GO back to the page to update the content" if you want to change the settings;
5. After all the leaf nodes are saved, a final confirmation page will be displayed. You can either choose to update the decision tree permanently, or discard the change.

CHAPTER SEVEN

CONCLUSIONS AND FUTURE DIRECTION

7.1 Conclusions

Sequence similarity search has been widely used by biologists in discovering functions, structure, and biochemical properties of novel biological sequences. An efficient similarity search relies on the choice of tools and their associated subprograms and numerous parameter settings. This could be very challenging for similarity search users, especially those at the beginner level. To assist researchers in selecting optimal programs and parameter settings, we have developed a web-based expert system, Smart Sequence Similarity Search (S^4).

S^4 is a Web-based expert system on biological sequence similarity search implemented in Java and Jess. It aims to help biologists choose optimal search program and corresponding parameters while conducting sequence similarity search. The project uses Jess as its reasoning core. It consists of four basic elements, including inference engine, knowledge base, database in XML format and user interface. Except for inference engine, which utilizes Jess expert shell, all the other three elements are developed in our project.

Knowledge base includes both rules and facts for knowledge process. Knowledge on similarity search is acquired from experts as well as literatures including user manuals provided by the well-known online services. These services considered in the project are BLAST search from NCBI, FASTA from EBI and Smith-waterman search from BBDJ.

Decision tree with four types of nodes is chosen as an ideal data structure to store the knowledge. The knowledge is initially stored in XML format, considering its clear tree-structure data representation and high readability suitable for biologists to update knowledge directly. It is parsed into Java objects and asserted into Jess expert system mainly in the form of facts. The only exception is the part of the knowledge on automatic option selection for auto decision node, which is defined as rules in the expert system. Most rules defined in Jess scripting language are responsible for processing decision tree traverse and extension.

User interfaces are generated by servlet or JSP. It utilizes a different thread from Jess expert system. Message passing between servlet and Jess expert system is not only responsible for message exchange between the two threads, but also for their synchronization. Besides, direct data access via Java object is also used for sharing information between the two threads.

A powerful and useful functionality of the S^4 system is its expandability. To allow easy improvement of the S^4 expert system, functions and interfaces for incorporating advanced knowledge from expert users are also provided. An expert user can extend the decision tree with more advanced knowledge in the form of subtree and make it take effect immediately. This allows our system to provide more accurate services to the users.

The project uses Apache Tomcat as Web server and has been tested thoroughly with different sequence (DNA or protein) inputs and search options. The suggested settings are consistent with currently-acquired knowledge. System improvement functions are also tested by repeating the service after the knowledge improvement. Besides, error and exception handling are considered under different conditions.

Our Jess-expert-system based S^4 project provides a solid advising tool for biologists to conduct efficient sequence similarity search. It has several features as listed below:

1. Platform-independent. Our system can be run on any web server supporting Java;

2. User friendly interface and easy access. Our system is Web-based, hence can be accessed from computers with Internet browser;

3. Easy-to-follow questions and options. All the sequence similarity knowledge are converted to easy-to-follow questions and options, allowing any researchers with basic biological knowledge to conduct sequence similarity search;

4. Comprehensive expert knowledge in sequence similarity search. The expert knowledge was obtained from thorough study of the servers and available literatures and contains the most up-to-date knowledge;

5. Knowledge expandability. The expert knowledge can be easily expanded if further improvements on the servers or algorithms are introduced;

6. User-defined knowledge. User-specific expert knowledge can be created by experts allowing more personalized searches.

7.2 Future Direction

Our project is an initial version to provide a Web-based expert system on biological knowledge. It can be improved in several aspects.

First of all, our project only supports interface for decision tree extension, which means that only the leaf nodes can be replaced by a new sub-tree. Moreover, the user is not allowed to modify the tree content any more once the information is added into the tree permanently except by directly editing the XML file that storing the information. Thus, user interfaces for updating the whole decision tree can make the project more powerful. Moreover, tools for decision tree visualization or navigation is also considered useful for better understanding the existing decision tree. A better interface can also be provided for the users to update the decision tree while navigating it. However, displaying such a big tree is technically very challenging. Users can also have their own decision trees so that the search is more personalized if user account is provided by our system. It will also prevent the decision tree from becoming erroneous if users input conflicting knowledge while updating the decision tree.

Secondly, we currently use XML file to store all the knowledge. This was chosen because of the high readability of XML file, which allows the biologists to edit the decision tree by directly modifying the XML code by simply using a text file editor. If proper decision tree edit interfaces are implemented, relational database, which is considered as a more efficient knowledge storage platform, should be added into the system, while the existing XML format can be used in data exchange.

Thirdly, improvement on the similarity search can also make our system more powerful and user-friendly. Currently, after optimal parameter settings are provided to the users, the actual similarity search is done on the remote server in a new window. Improvement can be done to submit the search and retrieve the result which will be displayed to the users in our system. This will allow the users to conduct the sequence similarity search in our system from start to end. Our system can also be improved to allow the users to provide their feedbacks based on the search results and modify the parameter settings based these feedbacks.

Finally, our expert knowledge is represented in the form of a decision tree, which can grow very fast when more nodes are added. Large decision tree can attenuate the system performance significantly. Novel knowledge-based structures should be developed to allow better performance.

BIOBLIOGRAPHY

[B1] Smith, T. F. and Waterman, M. S., "Identification of common molecular subsequences," *J.Mol.Biol.*, vol. 147, no. 1, pp. 195-197, Mar.1981.

[B2] Pearson, W. R. and Lipman, D. J., "Improved tools for biological sequence comparison," *Proc.Natl.Acad.Sci.U.S.A*, vol. 85, no. 8, pp. 2444-2448, Apr.1988.

[B3] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J., "Basic local alignment search tool," *J.Mol.Biol.*, vol. 215, no. 3, pp. 403-410, Oct.1990.

[B4] Chao, K. M., Pearson, W. R., and Miller, W., "Aligning two sequences within a specified diagonal band," *Comput.Appl.Biosci.*, vol. 8, no. 5, pp. 481-487, Oct.1992.

[B5] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J., "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389-3402, Sept.1997.

[B6] Pearson, W. R., "Comparison of methods for searching protein sequence databases," *Protein Sci.*, vol. 4, no. 6, pp. 1145-1160, June1995.

[B7] Rognes, T. and Seeberg, E., "Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors," *Bioinformatics.*, vol. 16, no. 8, pp. 699-706, Aug.2000.

[B8] Henikoff, S. and Henikoff, J. G., "Performance evaluation of amino acid substitution matrices," *Proteins*, vol. 17, no. 1, pp. 49-61, Sept.1993.

[B9] Vingron, M. and Waterman, M. S., "Sequence alignment and penalty choice. Review of concepts, case studies and implications," *J.Mol.Biol.*, vol. 235, no. 1, pp. 1-12, Jan.1994.

[B10] Altschul, S. F., "Amino acid substitution matrices from an information theoretic perspective," *J.Mol.Biol.*, vol. 219, no. 3, pp. 555-565, June1991.

[B11] Pearson, W. R., "Using the FASTA program to search protein and DNA sequence databases," *Methods Mol.Biol.*, vol. 24 pp. 307-331, 1994.

[B12] Giarratano, J. C. and Riley, G., *Expert Systems: Principles and Programming*, 3rd ed. 1998.

[B13] Shortliffe, E. H., Davis, R., Axline, S. G., Buchanan, B. G., Green, C. C., and Cohen, S. N., "Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the MYCIN system," *Comput.Biomed.Res.*, vol. 8, no. 4, pp. 303-320, Aug.1975.

[B14] Aikins, J. S., Kunz, J. C., Shortliffe, E. H., and Fallat, R. J., "PUFF: an expert system for interpretation of pulmonary function data," *Comput.Biomed.Res.*, vol. 16, no. 3, pp. 199-208, June1983.

[B15] Barker, V. E. and Oconnor, D. E., "Expert Systems for Configuration at Digital - Xcon and Beyond," *Communications of the Acm*, vol. 32, no. 3, pp. 298-317, 1989.

[B16] Bonissone, P. P. and Johnson, H. E., "Expert System for Diesel Electric Locomotive Repair (Reprinted)," *Human Systems Management*, vol. 4, no. 4, pp. 255-262, 1984.

[B17] Smith, D. H., Gray, N. A. B., Nourse, J. G., and Crandell, C. W., "Applications of Artificial-Intelligence for Chemical Inference .38. the Dendral Project - Recent Advances in Computer-Assisted Structure Elucidation," *Analytica Chimica Acta-Computer Techniques and Optimization*, vol. 5, no. 4, pp. 471-497, 1981.

[B18] IEEE Std.830-1998. IEEE Recommended Practice of Software Requirements Specifications. 1998.

[B19] Shpaer, E. G., Robinson, M., Yee, D., Candlin, J. D., Mines, R., and Hunkapiller, T., "Sensitivity and selectivity in protein similarity searches: a comparison of Smith-Waterman in hardware to BLAST and FASTA," *Genomics*, vol. 38, no. 2, pp. 179-191, Dec.1996.

[B20] Agarwal, P. and States, D. J., "Comparative accuracy of methods for protein sequence similarity search," *Bioinformatics.*, vol. 14, no. 1, pp. 40-47, 1998.