

DESIGN & IMPLEMENTATION OF A PIPELINE FOR HIGH-THROUGHPUT  
ENZYME FUNCTION PREDICTION

by

Seth Johnson  
A Thesis  
Submitted to the  
Graduate Faculty  
of  
George Mason University  
In partial fulfillment of  
The Requirements for the Degree  
of  
Master of Science  
Bioinformatics

Committee:

\_\_\_\_\_ Director

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_ Department Chairperson

\_\_\_\_\_ Dean, College of Science

Date: \_\_\_\_\_ Fall Semester 2006  
George Mason University  
Fairfax, Virginia

Design & Implementation of A Pipeline for High-Throughput Enzyme Function  
Prediction

A thesis submitted in partial fulfillment of the requirements for the degree of Master of  
Sciences at George Mason University

By

Seth Johnson  
Bachelor of Science  
University of Florida, 2001

Bachelor of Arts  
University of Florida, 1998

Bachelor of Science  
University of Florida, 1998

Director: Donald Seto, PhD  
Department of Bioinformatics

Fall Semester 2006  
George Mason University  
Fairfax, VA

## Acknowledgments

The author wishes to express sincere appreciation to Dr. Nela Zavaljevski for her assistance in the preparation of this manuscript. In addition, special thanks to Dr. Chenggang Yu whose familiarity with the needs and ideas of the project was helpful during the programming phase of this undertaking.

## Table of Contents

	Page
<b>ABSTRACT</b> .....	<b>X</b>
<b>INTRODUCTION</b> .....	<b>1</b>
NEED FOR HIGH-THROUGHPUT (HT) ANNOTATION TO SUPPORT HT SEQUENCING .....	1
CHALLENGES IN PROTEIN FUNCTION ANNOTATION .....	3
BACKGROUND AND RELATED WORK .....	5
<i>Overview of Existing Approaches for Enzyme Function Prediction</i> .....	5
<i>Overview of Related Work at BHS.AI</i> .....	10
<b>MATERIALS &amp; METHODS</b> .....	<b>15</b>
IMPROVEMENTS TO EXISTING EPD PIPELINE.....	15
<i>Combined Sequence Homology and Functionally Discriminative Site Search</i> .....	15
<i>Extending Prediction to Other Protein Function</i> .....	15
THE GENE ONTOLOGY (GO) CROSS-REFERENCE DATABASE .....	17
CSA WEB SCRAPER MODULE.....	24
INCORPORATING FDS & ACTIVE SITE INFORMATION TO SUPPLEMENT EPD.....	31
<i>3-digit EC Predictions</i> .....	31
<i>4-digit EC Predictions</i> .....	34
IMPLEMENTATION .....	36
<i>Software Package</i> .....	36
<i>High-Performance Computing Details</i> .....	36
<i>Serial &amp; Parallel Execution Time</i> .....	37
<i>Generated Database</i> .....	38
ENZYME VALIDATION SET.....	40
EVALUATION PARAMETERS .....	41
<i>Accuracy</i> .....	41
<i>Coverage</i> .....	42
<b>RESULTS &amp; DISCUSSION</b> .....	<b>43</b>
3-DIGIT EC FAMILIES.....	43
DISCUSSION OF 3-DIGIT RESULTS.....	47
4-DIGIT EC FAMILIES.....	49
DISCUSSION OF 4-DIGIT RESULTS.....	53
CONCLUSIONS & FUTURE WORK.....	55
<b>DISCLAIMER</b> .....	<b>58</b>
<b>APPENDIX A</b> .....	<b>59</b>
<b>APPENDIX B PART I</b> .....	<b>61</b>
<b>APPENDIX B PART II</b> .....	<b>62</b>

APPENDIX C.....	63
APPENDIX D.....	69
APPENDIX E.....	72
APPENDIX F .....	75
APPENDIX G.....	76
BIBLIOGRAPHY.....	81
BIBLIOGRAPHY.....	82
CURRICULUM VITAE.....	84

## List of Tables

	Page
TABLE 1. EFFECT OF INCREASE IN STRINGENCY OF FILTERS ON ACCURACY AND COVERAGE OF 3-DIGIT FAMILY PREDICTIONS. OPTIMUM BALANCE BETWEEN ACCURACY AND COVERAGE IS HIGHLIGHTED IN YELLOW (~94%).....	45
TABLE 2. EFFECT OF INCREASE IN STRINGENCY OF FILTERS ON ACCURACY AND COVERAGE OF 4-DIGIT FAMILY PREDICTIONS. OPTIMUM BALANCE BETWEEN ACCURACY AND COVERAGE IS HIGHLIGHTED IN YELLOW. INCREASE BY 9% IN ACCURACY IS ACHIEVED BY 1% DECREASE IN COVERAGE. ....	51

## List of Figures

	Page
FIGURE 1. ONE OF THE BEST-CHARACTERIZED EXAMPLES OF A MECHANISTICALLY DIVERSE ENZYME SUPERFAMILY IN SFLD IS THE ENOLASE SUPERFAMILY (ES) ILLUSTRATING CONNECTION BETWEEN CATALYZED REACTIONS AND COMMONALITY OF STRUCTURE.....	8
FIGURE 2. SUBFAMILY STRUCTURE COMPARISONS BETWEEN NON-OVERLAPPING EFICAZ (LEFT) AND OVERLAPPING EPD (RIGHT). .....	11
FIGURE 3. GENERATION OF 3-D & 4-D FAMILIES AND CORRESPONDING PROFILES FOR EPD. ....	12
FIGURE 4. CLUSTALW ALIGNMENT FOR 1.1.1.6 GLDA_BACST SUBFAMILY MEMBERS ILLUSTRATION HIGH DEGREE OF SEQUENCE CONSERVATION BETWEEN MEMBERS. ....	13
FIGURE 5. EQUATION FOR OBTAINING A NORMALIZED SCORE FROM A RAW SCORE BASED ON THRESHOLD VALUE OF THE MATCHING FAMILY.....	14
FIGURE 6. DIAGRAM OF IMPROVEMENTS TO CURRENT EPD PIPELINE. THEIR INTEGRATION WITH FDR, CREATION OF PARALLEL SEARCH USING RPS-BLAST AND CDD, AND CROSS-REFERENCING RESULTS USING GO MAPPING DATABASE. ....	16
FIGURE 7. GO MAPPINGS FILE FORMAT THAT HAS EXTERNAL DATABASE IDENTIFICATION ON THE LEFT AND GO TERMINOLOGY ON THE RIGHT OF '>' SEPARATOR. ....	18
FIGURE 8. AN EXCERPT FROM PFAM TO GO MAPPING FILE SHOWING EXAMPLES OF CONTAINED DATA. ....	18
FIGURE 9. DIAGRAM OF THE SUB-PIPELINE TO PARSE INDIVIDUAL GO MAPPING TEXT FILES. ....	20
FIGURE 10. GO CROSS-REFERENCE DATABASE SCHEMA SHOWING PATHS AND JOIN TABLES BETWEEN DIFFERENT EXTERNAL DATABASES. ....	21
FIGURE 11. WEB GRAPHIC USER INTERFACE FOR THE GO MAPPINGS DATABASE. ....	22
FIGURE 12. LIST PAGE OF CSA LITERATURE BASED ENTRIES CONTAINING PDB NOTATION WITH LINK TO DETAILS PAGE (ORANGE OVALS), NAME OF AN ENZYME, EC NUMBER, AND CATH CODE. ....	26
FIGURE 13. SCHEMATIC DIAGRAM OF CSA LITERATURE BASED ENTRIES LIST WEB SCRAPER PROCESS FLOW. ....	27
FIGURE 14. TYPICAL CSA ENTRY DETAIL PAGE WITH LITERATURE CATALYTIC SITES INFORMATION. DATA NEED FOR PIPELINE IS MARKED BY ORANGE OVALS. ....	28
FIGURE 15. FLOW CHART OF LOGIC IN CSA & SWISS-PROT CATALYTIC SITE ENTRY PAGE WEB SCRAPER. ....	30
FIGURE 16. FDS PROFILE FOR DHB7_HUMAN SUBFAMILY FEATURING ACCURACY, SENSITIVITY, SPECIFICITY PARAMETERS FOR THE PROFILE AND FOR EACH ACTIVE SITE POSITION.....	32
FIGURE 17. SEVERAL PREDICTIONS OF AN UNKNOWN PROTEIN NAMED ON THE LEFT TO SEVERAL EC NUMBERS AND SUBFAMILIES IN THE MIDDLE. EACH SUBFAMILY THRESHOLD IS OUTLINED IN RED WITH THE RAW SCORE NEXT TO IT AS THE LAST COLUMN.....	33
FIGURE 18. FLOW CHART DIAGRAM OUTLINING MAJOR SUBUNITS AND DIRECTION OF INFORMATION PASSAGE IN "EPD + FUNCTIONAL SITES" ENZYME FUNCTION PREDICTION PIPELINE. ....	34
FIGURE 19. FLOW CHART DIAGRAM OUTLINING A CONCEPT OF PARALLEL EXECUTION OF A PIPELINE. ....	38
FIGURE 20. FINAL DATABASE SCHEMA CONTAINING FDS INFORMATION HIGHLIGHTED IN ORANGE. ....	39
FIGURE 21. EQUATION FOR CALCULATING ACCURACY BASED ON THE NUMBER OF CORRECT PREDICTIONS (TP).....	41
FIGURE 22. EQUATION FOR CALCULATING INPUT COVERAGE BASED ON FRACTION OF INPUT PROTEINS WITH ACCEPTABLE PREDICTIONS. ....	42

FIGURE 23. NMS vs. SCF PLOT. DISTINCT CLUSTERING OF TP (TURQUOISE) AND FP (BURGUNDY) PREDICTIONS IN DIFFERENT REGIONS OF THE PLOT ALLOWS APPLICATION OF FILTERING (COLORED RECTANGLES) TO REMOVE FP AND INCREASE ACCURACY. ....	44
FIGURE 24. ACCURACY (BLUE) vs. COVERAGE (PINK) PLOT INDICATING THAT OPTIMUM TRADE OFF BETWEEN THEM OCCURS DURING APPLICATION OF FILTER #6 (GREEN DOUBLE ARROW) WHEN VALUES HAVING $NMS < 0.7$ AND $SCF < 0.5$ ARE REJECTED. ....	46
FIGURE 25. PLOT OF ACCURACY & COVERAGE vs. SEQUENCE IDENTITY INDICATING A DIRECT RELATIONSHIP BETWEEN THESE VALUES. AT 60% SEQUENCE IDENTITY ACCURACY OF PREDICTION IS JUST UNDER 90% (SHOWN BY GREEN DOUBLE ARROWS). ....	48
FIGURE 26. NMS vs. SCF FOR 4-DIGIT EC FAMILY PREDICTIONS INDICATING WEAK CLUSTERING OF TP AND FP PREDICTIONS. ORANGE RECTANGLE INDICATES THE MOST STRINGENT FILTER APPLIED TO PREDICTIONS.....	50
FIGURE 27. ACCURACY vs. COVERAGE PLOT OF 4-DIGIT FAMILIES PREDICTIONS. OPTIMUM TRADE OFF BETWEEN ACCURACY AND COVERAGE OCCURS WITHOUT APPLICATION OF FILTERS. ....	52

## List of Abbreviations

**2NF** – Second Normal Form.

**BHSAI** - Biotechnology High Performance Computing Software Applications Institute.

**BLAST** – Basic Local Alignment Search Tool.

**BST** – Binary Search Tree.

**CDD** – Conserved Domain Database.

**CHIEFc.** - Conservation-controlled HMM Iterative procedure for Enzyme Family classification.

**ClustalW** - a general purpose program that identifies alignments within multiple DNA or protein sequences.

**COG** – Clusters of Orthologous Groups.

**CSA** – Catalytic Site Atlas.

**CSV** – Comma Separated Values

**DNA** – Deoxyribonucleic Acid.

**DoD** – Department of Defense

**EC number** – Enzyme Commission number.

**EF** – Evolution Footprinting.

**EFICAz** - Enzyme Function Inference by Combined Approach.

**EPD** - The Enzyme Profile Database.

**FDR** – Functionally Discriminating Residues.

**FDS** – Functionally Discriminating Sites.

**GO** – Gene Ontology.

**GUI** – Graphic User Interface.

**HMM** – Hidden Markov Model.

**HTML** – Hyper Text Markup Language.

**IDE** – Integrated Development Environment.

**NMR** – Nuclear Magnetic Resonance.

**NMS** – Normalized Maximum Score.

**ORF** – Open Reading Frame.

**PHP** – PHP: Hypertext Preprocessor.

**PRIAM** - Profiles pour l'identification automatisee du metabolisme.

**PSI** - Protein Structure Initiative.

**PSI-BLAST** – Position-Specific Iterative BLAST.

**RDBMS** – Relational Database Management System.

**RPS-BLAST** – Reverse Position Specific BLAST.

**SCF** – Site Conservation Fraction.

**SFLD** - Structure-Function Linkage Database.

# **ABSTRACT**

## **DESIGN & IMPLEMENTATION OF A PIPELINE FOR HIGH-THROUGHPUT ENZYME FUNCTION PREDICTION**

Seth Johnson, M.S.

George Mason University, 2006

Thesis Director: Dr. Donald Seto

The goal of this research is the integration of the function prediction modules into a pipeline that will run on a high performance computing platform and which will enable enzyme function prediction on a proteomic level. Enzyme families developed for Enzyme Profile Database (EPD) will be supplemented by a relational database, which will encompass the information about conserved and predicted Functionally Discriminating Residues (FDR) and Functionally Discriminative Sites (FDS). In addition, information about enzyme function will be collected from Swiss-Prot, InterPro, and Gene Ontology (GO) databases, and stored into appropriate fields in the database. Automated mechanism for periodical updates of the database from the Swiss-Prot, InterPro, and GO will be provided. The EPD and enzyme function predictions based on it will be integrated into the pipeline and supplemented with FDR information. Results will be cross-referenced with predictions based on Swiss-Prot, InterPro, GO, and Conserved Domain Database (CDD). Bioinformatics tools will be also developed to automate the search for active sites and FDR's from Swiss-Prot and Catalytic

Site Atlas (CSA). A research contribution of this project will be the comparative analysis and implementation of two algorithms for functionally determining site search:

1. A fast “greedy search” method that implements simple residue conservation measures.
2. A novel method based on a classification algorithm and improved conservation measures for function discrimination.

The results of this analysis will be implemented in a software module for confidence evaluation of predicted enzyme functions and integrated in the function prediction pipeline. Performance will be assessed on a selected set of pre-annotated enzyme protein sequences from various pathogenic bacteria.

## Introduction

### ***NEED FOR HIGH-THROUGHPUT (HT) ANNOTATION TO SUPPORT HT SEQUENCING***

Rapid advancements in high-throughput DNA sequencing created an ever widening rift between a number of identified genes and their annotated function. Modern 96-Capillary DNA Sequencers from Applied Biosystems are capable of sequencing up to 2,000,000 bases per day each with little or no human control. On the other hand, function annotation for the newly sequenced Open Reading Frames (ORF) lags far behind. Structure-based annotation methods require a solved 3-dimensional protein structure that is derived through X-ray crystallography and NMR technologies. Recently, a Protein Structure Initiative (PSI) has been announced in the emerging field of structural genomics. The PSI is a large-scale, high-throughput effort to increase the number of structures of unique, non-redundant proteins, permitting the study of a broad range of protein structures. This will be accomplished by a coordinated effort of federal, university, and industrial scientists to reduce costs, increase efficiency and success rates, and decrease the time for the production and determination of the three-dimensional atomic-level structures of proteins (2). On the other hand, sequence-based annotation methods would offer a solution that would be available on the fly as soon as the complete sequence for the ORF is available. This purely

computational method enables researchers to bypass the expensive and time consuming purification and crystallization necessary for exact 3-dimensional structure resolution. However, it was recognized that sequence information must be enlarged using various sources of proteomic information, such as protein interactions, microarray data for function recognition, etc.

## ***CHALLENGES IN PROTEIN FUNCTION ANNOTATION***

Characterizing the function of proteins found in pathogenic organisms facilitates development of effective therapeutics and prophylactic measures against the illicit effects of the pathogen. However, the amount of data resulting from the rapid increase in newly identified protein sequences due to advances in sequencing limits the ability to experimentally determine the function of every one of these novel proteins. Bioinformatics algorithms and software packages that utilize a combination of available sequence and structural information for protein function prediction are under active development (3). These include modification of select algorithms, integration of disparate software tools/databases into a single software package, continuous maintenance, and updating of this software for select high performance software platforms. Complex and vaguely established relationship between sequence, structure, and function of well characterized proteins have spawned several avenues of approach to function prediction. Numerous databases house results of those computational approaches, many of which are erroneous and/or require human curation in order to correct inaccuracies as methods improve and more information become available. Thus, there exists a need for reliable high-throughput function prediction platform to complement high-throughput sequencing.

An integrated system for protein function prediction at the proteomic level is under development at the DoD Biotechnology High Performance Computing Software Applications Institute (BHS AI). The system consists of three modules:

1. A sensitive homology search method based on profiles derived from protein families (1).
2. Enzyme Profile Database (EPD), a customized database under development at BHS AI.
3. Information about enzyme active sites and other functionality determining sites collected from external databases or computed from the customized database, which provides additional evidence for homology-based function prediction.

Prior to this research, these modules were used independently, and results were combined manually to provide hypotheses of protein function for specified protein sequences. Lack of automated means of combining these modules precluded their usage in high throughput pipeline.

## ***BACKGROUND AND RELATED WORK***

### **Overview of Existing Approaches for Enzyme Function Prediction**

#### HOMOLOGY-BASED PREDICTION

There are many ways to approach protein function prediction. This work will focus mainly on a homology-based approach. The assumption behind the methodology is that evolutionary related proteins (homologs) share function. Homologous proteins arise from mutations in a common ancestor coding gene. Through the process of gene divergence, some gene mutations have been accepted by natural selection because they preserved the folding and function of the coded protein. Generally, homology-based type of enzyme function prediction is divided into two broad categories:

1. Sequence-based Methods: EFICAz (4), PRIAM (5). These methods usually employ one or several basic approaches for enzyme function prediction based on (i) homology transfer, (ii) presence of a pattern or motif, or (iii) identification of functional residues.
  - a. EFICAz (Enzyme Function Inference by Combined Approach) is an automatic engine for large-scale enzyme function inference that combines predictions from four different methods developed and optimized to achieve high prediction accuracy:

- (i) Recognition of Functionally Discriminating Residues (FDRs) in enzyme families obtained by a clustering method called Conservation-controlled HMM Iterative procedure for Enzyme Family classification (CHIEFc) (4). CHIEFc groups proteins together into families based on a minimum of 30% sequence identity cutoff. FDRs are determined using Multiple Sequence Alignment (MSA) of family members.
- (ii) Pair-wise sequence comparison using a family specific Sequence Identity Threshold.
- (iii) Recognition of FDRs in multiple Pfam enzyme families.
- (iv) Recognition of multiple Prosite patterns of high specificity.

This method defines each enzyme family as a group of proteins that are evolutionarily related and share four or three digits of their EC numbers. It applies an 'Evolutionary Footprinting' (EF) approach to identify residues in an enzyme family that can discriminate between sequences having the specified EC number and those sequences with other functions (different EC numbers or non-enzymes). It is an approach to finding functionally important sequences in the genome that relies on detecting their high degrees of conservation across different species. Three-dimensional structure information is not required in any stage of the method.

- b. PRIAM is a method for automated enzyme detection in a fully sequenced genome, based on the classification of enzymes in the ENZYME database (13). PRIAM relies on sets of position-specific scoring matrices ('profiles')

automatically tailored for each ENZYME entry. Automatically generated logical rules define which of these profiles is required in order to infer the presence of the corresponding enzyme in an organism. The methodology behind profile creation can be adjusted for other groups of proteins (subfamilies).

2. Structure-based Methods: CSA (6), SFLD (7). These resources center on solved 3-dimensional structure of the enzyme molecules and use it to mine information and correlate data.
  - a. The Catalytic Site Atlas (CSA) is a database documenting enzyme active sites and catalytic residues in enzymes of 3D structure. It defines a classification of catalytic residues which includes only those residues thought to be directly involved in some aspect of the reaction catalyzed by an enzyme. These residues can be directly used as FDRs during enzyme function prediction.
  - b. The Structure-Function Linkage Database (SFLD) is a tool for the investigation of protein sequence, structure, and function. It links evolutionary related sequences and structures from mechanistically diverse superfamilies of enzymes to their chemical reactions and correlates conserved active site residues with specific partial reactions that all members of a superfamily perform. It was developed by the Babbitt Laboratory in collaboration with the UCSF Resource for Biocomputing, Visualization, and Informatics. The main aim of SFLD project is to provide a more reliable nomenclature for enzyme function at the level of partial chemical reactions

since it was recognized that EC classification is not always accurate. Figure 1 contains an example of the principle behind SFLD.

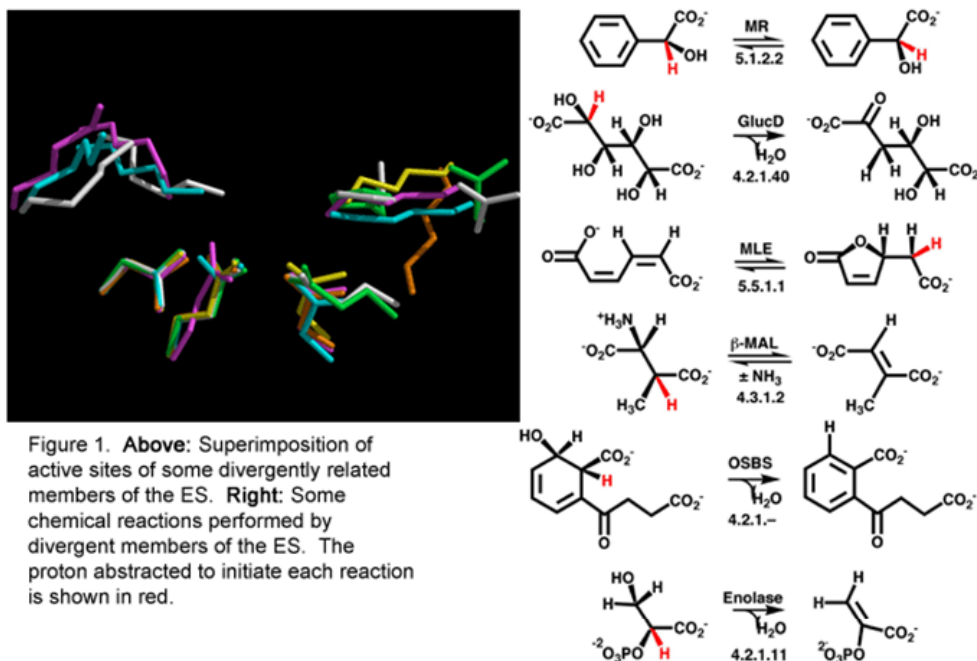


Figure 1. **Above:** Superimposition of active sites of some divergently related members of the ES. **Right:** Some chemical reactions performed by divergent members of the ES. The proton abstracted to initiate each reaction is shown in red.

**Figure 1.** One of the best-characterized examples of a mechanistically diverse enzyme superfamily in SFLD is the enolase superfamily (ES) illustrating connection between catalyzed reactions and commonality of structure.

## OTHER SOURCES OF FUNCTION CLASSIFICATION

Other sources of protein and enzyme functional classification include GO and COG. The Gene Ontology (GO) Project can be split broadly into two parts. The first is the ontology itself, a controlled vocabulary of terms split into three related ontologies covering basic areas of Molecular Biology: the molecular function of gene products, their role in multi-step

biological processes, and their localization to cellular components. The ontology is continuously updated, and new snapshot releases are made available on a monthly basis. The second part of GO is annotation, the characterization of gene products using terms from the gene ontology. The members of the GO Consortium make their data publicly available through the GO website (8).

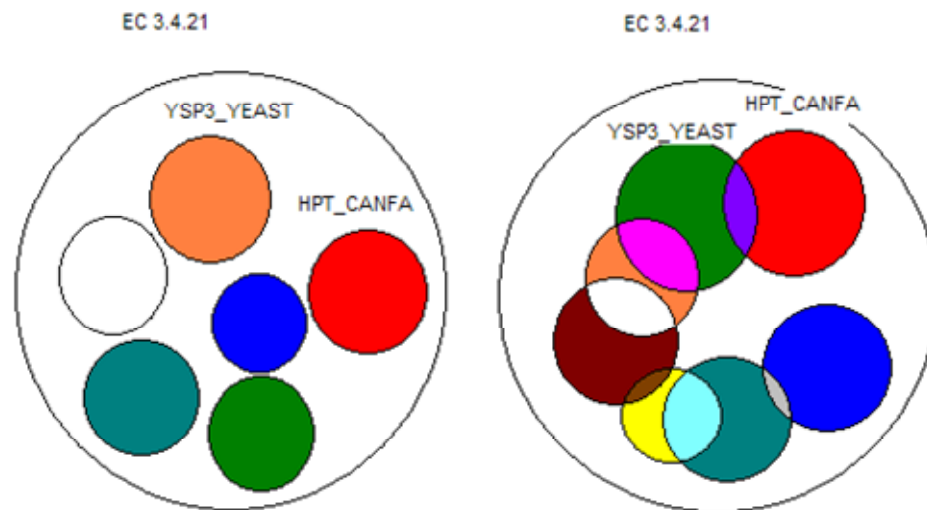
Comparison of proteins encoded in 66 complete genomes from 14 major phylogenetic lineages and elucidation of consistent patterns of sequence similarities allowed the delineation of 4872 Clusters of Orthologous Groups (COGs). Such classifications are based on two fundamental notions from evolutionary biology: orthology and paralogy, which describe the two fundamentally different types of homologous relationships between genes. Orthologs are homologous genes derived by vertical descent from a single ancestral gene in the last common ancestor of the compared species. Paralogs, in contrast, are homologous genes, which, at some stage of evolution of the respective gene family, have evolved by duplication of an ancestral gene. The underlying premise is that orthologs are more similar to each other than they are to any other protein from the respective genomes. Each COG is assumed to have evolved from an individual ancestral gene through a series of speciation and duplication events. This relation automatically yields a number of functional predictions for poorly characterized genomes. The COGs comprise a domain/motif framework for functional and evolutionary genome analysis (9).

## **Overview of Related Work at BHSAI**

A novel combined approach, Enzyme Profile Database (EPD), is under development at BHSAI to facilitate function annotation for enzymatic proteins. EPD implements sequence based function prediction similar to EFICAz and PRIAM methods. However, it differs from them in several key areas.

### **OVERLAPPING SUBFAMILIES**

EPD, unlike EFICAz and PRIAM, implements overlapping subfamilies for each EC number where appropriate. The overlap allows for greater flexibility when one or more proteins are poised to fit more than one subfamily. Figure 2 illustrates a comparison between non-overlapping and overlapping families.



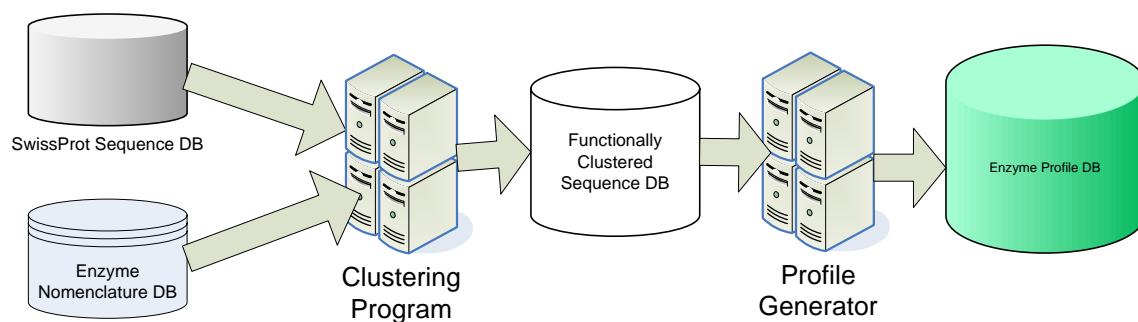
**Figure 2.** Subfamily structure comparisons between non-overlapping EFICAz (left) and overlapping EPD (right).

1. Families of enzymes are based on ENZYME database (release 37) (13). Each family consists of a group of enzymes that share either 3- or 4-digit EC notation.
2. Each family consists of a number of subfamilies clustered based on homologies using Position-Specific Iterative BLAST (PSI-BLAST) following the same general direction as indicated for PRIAM (5). PSI-BLAST searches a profile/motif against a database of sequences to find matches.
3. EFICAz, on the other hand, uses Hidden Markov Model (HMM) to group its subfamilies together. A hidden Markov model (HMM) is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters.

The extracted model parameters can then be used for pattern recognition applications. A HMM can be considered as the simplest dynamic Bayesian network. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states (11).

### ENZYME FAMILY DEFINITION

Enzyme families were defined using remote homology detection based on PSI-BLAST with improved profile estimation. These improvements were due to an additional step that verified family alignment using MSA tool ClustalW (12). Figure 4 shows clear signs that its members are closely related. It indicates that more than 40% of the residues in these sequences exhibit identity and about 70% shown to be at least semi-conserved. Creation of EPD families and outlining of discriminating profiles is outlined in Figure 3.



**Figure 3.** Generation of 3-d & 4-d families and corresponding profiles for EPD.



thresholds of E-value  $\approx 0.01$  for EFICAz and E-value  $\approx 10^{-10}$  for PRIAM. These thresholds, especially the latter one, are very stringent and do not reflect differences between families such as the size of family and degree of sequence identity between its members. In EPD, family thresholds are based on an extensive training set and specified prediction accuracy.

When an unknown sequence is searched against EPD, Normalized Score ( $T_N$ ) of the match is based on the Threshold Value ( $T_C$ ) and the Raw Score ( $T$ ). It is calculated according to the Figure 5 below.

$$T_N = \frac{T - T_C}{T_C}$$

**Figure 5.** Equation for obtaining a Normalized Score from a Raw Score based on Threshold Value of the matching family.

## GENOME-WIDE PREDICTION OF ENZYMES

Comprehensive nature of EPD profiles allows prediction of nearly all enzymes classified by 3- and 4-digit EC numbers. Predictions are unavailable only for families that contain too few members for profile estimation, typically less than three. Total number of families currently stands at 190 for 3-digit families with several subfamilies in each of them, and 1,747 4-digit families with at least one subfamily in each. Thus, there exists a possibility to predict all of the enzymes in a given organism. By identifying putative enzymes in translated ORFs of a newly sequenced unknown organism its nature and even identity can be outlined based on the presence of unique enzymes.

## **Materials & Methods**

### ***IMPROVEMENTS TO EXISTING EPD PIPELINE***

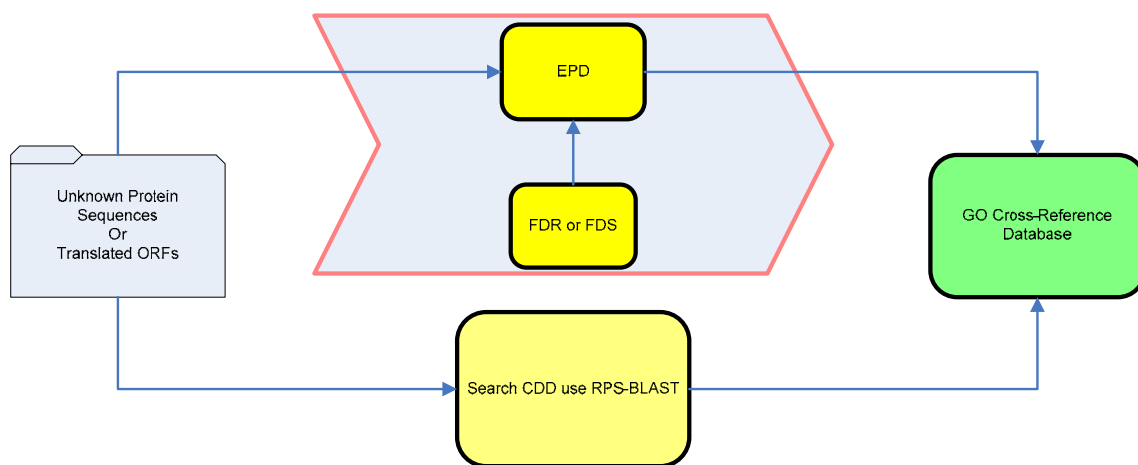
#### **Combined Sequence Homology and Functionally Discriminative Site Search**

An improvement upon homology-based methods of function predictions obtained from the EPD subroutine (sequence based predictions) is function prediction evaluation based on the Functionally Discriminative Sites (FDS) data. These sites are obtained by searching for the conserved sites in the sequence alignment of 3-digit families. For 4-digit families they are obtained from CSA and Swiss-Prot literature based active site databases. This approach has been previously shown to produce effective results (4,21). Positive re-enforcement of the sequence based prediction by active site information provides further evidence and support to initial guesses. This improvement is outlined by gray arrow with red edges on Figure 6.

#### **Extending Prediction to Other Protein Function**

In addition to EPD supported by FDS & active site data, Conserved Domain Database (CDD) is included to extend the score to other protein functions. Conserved domains are based on recurring sequence patterns or motifs. The un-curated section of CDD contains

domains imported from SMART, Pfam, and COGs. The source databases also provide descriptions and links to citations. Because conserved domains correspond to compact structural units, CDs are linked to 3D structure when possible. The NCBI-curated section of CDD attempts to group ancient domains related by common descent into family hierarchies. To identify conserved domains in a protein sequence, the CD-Search service uses Reverse Position-Specific BLAST algorithm (RPS-BLAST). The query sequence is compared to a position-specific score matrix prepared from the underlying conserved domain alignment. Hits may be displayed as pairwise alignments of the query sequence with representative domain sequences, or as MSA (10). Incorporation of CDD search is outlined by lower half of Figure 6.



**Figure 6.** Diagram of improvements to current EPD pipeline. Their integration with FDR, creation of parallel search using RPS-BLAST and CDD, and cross-referencing results using GO Mapping database.

## ***THE GENE ONTOLOGY (GO) CROSS-REFERENCE DATABASE***

The creation of an in-house database that provides a common reference point between numerous external databases is essential to combining predictions from several diverse sources such as CDD collection and Swiss-Prot notations. Biologists currently waste a lot of time and effort in searching for all of the available information about each small area of research. This is hampered further by the wide variations in terminology that may be common usage at any given time, which inhibit effective searching by both computers and people. For example, if you were searching for new targets for antibiotics, you might want to find all the gene products that are involved in bacterial protein synthesis, and that have significantly different sequences or structures from those in humans. If one database describes these molecules as being involved in 'translation', whereas another uses the phrase 'protein synthesis', it will be difficult for you - and even harder for a computer - to find functionally equivalent terms. The Gene Ontology (GO) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases (14). The three organizing principles behind GO database are **cellular component**, **biological process**, and **molecular function**. Currently, GO provides separate files that outline mappings of external classification systems to GO. These files contain annotations from external database systems, such as Enzyme Commission numbers, SWISS-PROT keywords and TIGR roles, indexed to equivalent GO terms. The mappings are typically made

manually, details can be found in the file header. The GO Mappings file format is outlined in Figure 7.

```
external system identifier: external system term name/id > GO:GO term name ; GO:id
```

**Figure 7.** GO Mappings file format that has external database identification on the left and GO terminology on the right of ‘>’ separator.

To make the cross-reference information in those files useful for high-throughput computing it is necessary to parse the files and import the relationships into Relational Database Management System (RDBMS). A snippet of real data from a Pfam2GO file is shown in Figure 8.

```
!date: 2006/07/17 22:07:14
!Mapping of Pfam entries to GO
!http://www.sanger.ac.uk/Software/Pfam/index.shtml
!Uses Interpro2Go by Nicola Mulder, Hinxton
!
Pfam:PF00001 7tm_1 > GO:rhodopsin-like receptor activity ; GO:0001584
Pfam:PF00001 7tm_1 > GO:G-protein coupled receptor protein signaling pathway ; GO:0007186
Pfam:PF00001 7tm_1 > GO:integral to membrane ; GO:0016021
Pfam:PF00002 7tm_2 > GO:G-protein coupled receptor activity ; GO:0004930
Pfam:PF00002 7tm_2 > GO:membrane ; GO:0016020
Pfam:PF00003 7tm_3 > GO:metabotropic glutamate, GABA-B-like receptor activity ; GO:0008067
Pfam:PF00003 7tm_3 > GO:membrane ; GO:0016020
Pfam:PF00004 AAA > GO:ATP binding ; GO:0005524
Pfam:PF00005 ABC_tran > GO:ATP binding ; GO:0005524
Pfam:PF00005 ABC_tran > GO:ATPase activity ; GO:0016887
Pfam:PF00006 ATP-synt_ab > GO:ATP binding ; GO:0005524
Pfam:PF00006 ATP-synt_ab > GO:hydrogen-transporting ATP synthase activity, rotational mechanism ; GO:0046933
Pfam:PF00006 ATP-synt_ab > GO:hydrogen-transporting ATPase activity, rotational mechanism ; GO:0046961
```

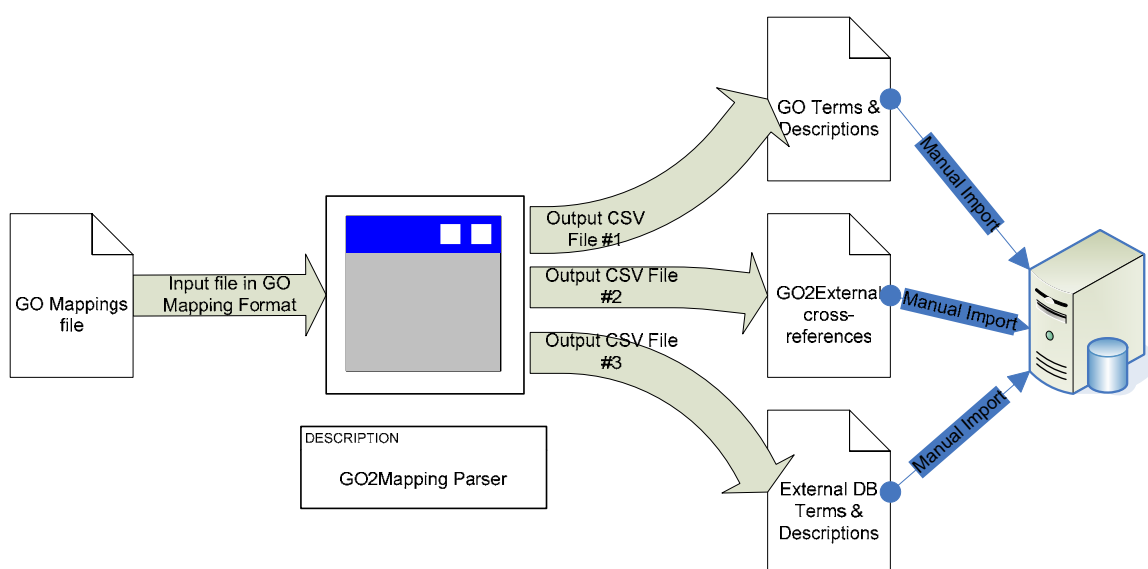
**Figure 8.** An excerpt from Pfam to GO mapping file showing examples of contained data.

Initially, cross-references to five external databases were included in the GO Mapping RDBMS:

1. Pfam - large collection of multiple sequence alignments and Hidden Markov Models covering many common protein domains and families (15).
2. COG - Clusters of Orthologous Groups of proteins (COGs) were delineated by comparing protein sequences encoded in complete genomes, representing major phylogenetic lineages. Each COG consists of individual proteins or groups of paralogs from at least 3 lineages and thus corresponds to an ancient conserved domain (9).
3. PROSITE - a database of protein families and domains. It consists of biologically significant sites, patterns and profiles that help to reliably identify to which known protein family (if any) a new sequence belongs (16).
4. ProDom - a comprehensive set of protein domain families automatically generated from the SWISS-PROT and TrEMBL sequence databases (17).
5. EC – provides common names of all listed enzymes, along with their EC numbers (18).

Source code for the GO Mapping files parser is located in Appendix A of the current document. It is implemented in Perl 5.8 programming language and takes one command line parameter, the name of the mapping file. Based on the data contained in the input file, software automatically determines which of the five mapping files has been supplied and proceeds accordingly. The output of the routine consists of three comma separated value (CSV) files that contain: (i) GO terms and corresponding descriptions, (ii) cross-references between GO terms and external database terms, and (iii) external database terms with corresponding descriptions. This information split was necessary to make the database conform to 2nd Normal Form (2NF). It requires that all data elements in a table are

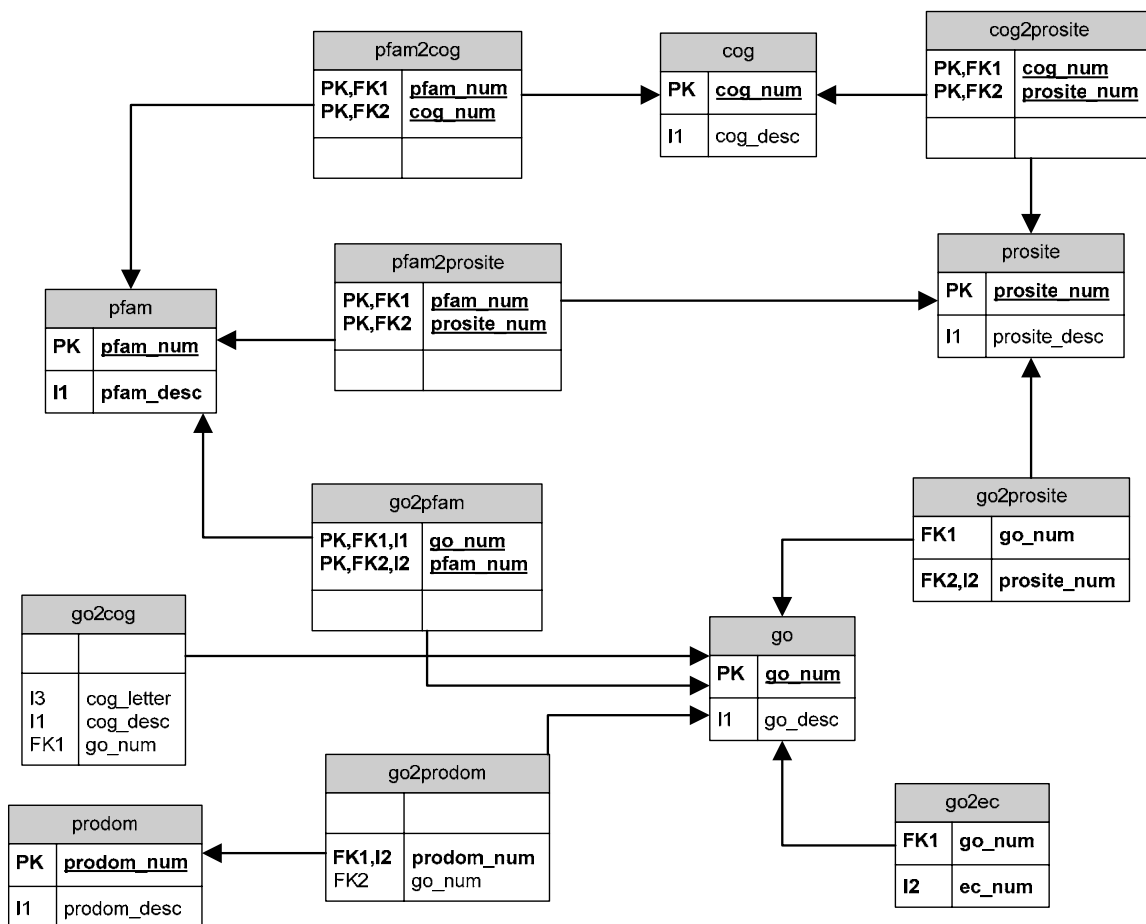
functionally dependent on the table's entire primary key (a value that can be used to identify a unique row in a table). If data elements only depend on part of a primary key, then they are parsed out to separate tables. If the table has a single field as the primary key, it is automatically in 2NF. The general flow of information during parsing of the GO Mapping files is outlined in Figure 9.



**Figure 9.** Diagram of the sub-pipeline to parse individual GO Mapping text files.

The entire cross-reference database that houses the information from GO Mapping files consists of 13 tables. Database schema is outlined in Figure 10. Several notes about the schema have to be clarified. There are two separate tables dealing with COG designations. In the course of the research it was determined that GO mapping to COG database is done only on the level of functional categories and not orthologous groups themselves.

Therefore, it was deemed necessary to include second table that would map COG notations to two other external databases. This mapping would allow a linkage to be created from each orthologous group to a GO term and beyond to other external database notations. This lightweight implementation of the cross-reference database anchored around GO terms was assessed to be important enough to warrant standalone implementation outside of the HT pipeline. Currently, a user has to search several different databases in order obtain the same information.



**Figure 10.** GO cross-reference database schema showing paths and join tables between different external databases.

A standalone web interface shown in Figure 11 was created using HyperText Markup Language (HTML) with embedded PHP in Macromedia DreamWeaver MX Integrated Development Environment (IDE). PHP stands for *PHP: Hypertext Preprocessor*. The confusion stems from the fact that the first word of the acronym is the acronym. This type of acronym is called a recursive acronym. PHP is a reflective programming language originally designed for producing dynamic Web pages. PHP is used mainly in server-side application software. The database and the front-end are hosted on author's private website through GoDaddy.com hosting service. It is free and available for general public to use. The URL is <http://binf.vsevolod.com/GO.php>.

GENE ONTOLOGY MAPPINGS																	
GO:	Pfam:	COG:	ProDom:	PROSITE:	EC:												
<table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr style="background-color: #d2b48c;"> <th style="width: 5%;">DB</th> <th style="width: 45%;">ID (0.0.0.0)</th> <th style="width: 50%;">Search</th> </tr> </thead> <tbody> <tr> <td>EC:</td> <td><input type="text" value="3.1.1.29"/></td> <td><input type="button" value="Submit"/></td> </tr> </tbody> </table>						DB	ID (0.0.0.0)	Search	EC:	<input type="text" value="3.1.1.29"/>	<input type="button" value="Submit"/>						
DB	ID (0.0.0.0)	Search															
EC:	<input type="text" value="3.1.1.29"/>	<input type="button" value="Submit"/>															
3.1.1.29																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #d2b48c;"> <th style="width: 15%;">GO ID</th> <th style="width: 20%;">GO Desc.</th> <th style="width: 15%;">Pfam ID</th> <th style="width: 50%;">Pfam Desc.</th> </tr> </thead> <tbody> <tr> <td><a href="#">0004045</a></td> <td>aminoacyl-tRNA hydrolase activity</td> <td><a href="#">PF01195</a></td> <td>Peptidyl-tRNA hydrolase (25-208)</td> </tr> </tbody> </table>				GO ID	GO Desc.	Pfam ID	Pfam Desc.	<a href="#">0004045</a>	aminoacyl-tRNA hydrolase activity	<a href="#">PF01195</a>	Peptidyl-tRNA hydrolase (25-208)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #d2b48c;"> <th style="width: 15%;">ProDom ID</th> <th style="width: 85%;">ProDom Desc.</th> </tr> </thead> <tbody> <tr> <td><a href="#">PD005324</a></td> <td>PeptRNAhydrolase</td> </tr> </tbody> </table>		ProDom ID	ProDom Desc.	<a href="#">PD005324</a>	PeptRNAhydrolase
GO ID	GO Desc.	Pfam ID	Pfam Desc.														
<a href="#">0004045</a>	aminoacyl-tRNA hydrolase activity	<a href="#">PF01195</a>	Peptidyl-tRNA hydrolase (25-208)														
ProDom ID	ProDom Desc.																
<a href="#">PD005324</a>	PeptRNAhydrolase																
				<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #d2b48c;"> <th style="width: 15%;">PROSITE ID</th> <th style="width: 85%;">PROSITE Desc.</th> </tr> </thead> <tbody> <tr> <td><a href="#">PS01195</a></td> <td>Peptidyl-tRNA hydrolase signature 1.</td> </tr> <tr> <td><a href="#">PS01196</a></td> <td>Peptidyl-tRNA hydrolase signature 2.</td> </tr> </tbody> </table>		PROSITE ID	PROSITE Desc.	<a href="#">PS01195</a>	Peptidyl-tRNA hydrolase signature 1.	<a href="#">PS01196</a>	Peptidyl-tRNA hydrolase signature 2.						
PROSITE ID	PROSITE Desc.																
<a href="#">PS01195</a>	Peptidyl-tRNA hydrolase signature 1.																
<a href="#">PS01196</a>	Peptidyl-tRNA hydrolase signature 2.																

©2005 Seth Johnson.  
All Rights Reserved.

**Figure 11.** Web graphic user interface for the GO Mappings database.

Graphic User Interface (GUI) of the Gene Ontology Mappings has the following properties and capabilities:

1. Centralized navigation menu bar that allows user to access one of the six pages in order to initiate a search query for “synonyms” for each particular database.
2. Search query window with *Submit* button featuring an example of the acceptable input ID format.
3. Results area that displays an ID submitted in red and several tables containing “synonyms” that were found from each cross-linked database. Each ID field of a “synonym” is a hyperlink, and when followed will provide “synonyms” for that particular ID.

## ***CSA WEB SCRAPER MODULE***

An important task performed in this research was the acquisition of literature based catalytic sites from Catalytic Site Atlas (CSA). One goal of this project was to determine whether EPD function predictions for 4-digit EC numbers can be further supported and fine-tuned by the data mined from CSA literature entries. The information about catalytic sites found in the database was shown previously to have a differentiating effect on enzyme function prediction and it was expected that it will help when EPD predictions are inconclusive.

One of the problems with acquisition of literature entries of CSA is the unavailability of database for public download in batches. Creators and curators of the database at EMBL-EBI have made it clear that they do not intend to provide complete database dump for download. Thus, the database is only available for human browsing and cannot be readily incorporated into a pipeline. CSA contains both literature and homology based information. In this project, only literature references curated in CSA are used.

In order to obtain the necessary information, a general concept known as **Screen Scraping** was applied. Screen scraping is a technique in which a computer program extracts data from the display output of another program. The program doing the scraping is called a screen scraper. The key element that distinguishes screen scraping from regular parsing is that the

output being scraped was nominally intended for human consumption, not machine interpretation (19). One of the flavors of screen scraping is **Web Scraping**. Web Scraping is different from Screen scraping in the sense that a Web Site is really not a screen, but a live HTML/JavaScript based application, with a graphics interface in front of it. Thus Web Scraping does not involve working at the visual interface as Screen Scraping, but rather Web Scraping works on the underlying object structure, Document Object Model (DOM) of the HTML and JavaScript (20).

First step is to obtain a complete list of PDB structures for which literature referenced catalytic sites are available. This list is available by accessing the following URL:

[http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/CSA/CSA\\_Browse.pl](http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/CSA/CSA_Browse.pl) . The

webpage shown in Figure 12 can be saved locally through a browser thereby providing a source for Web Scraper software.

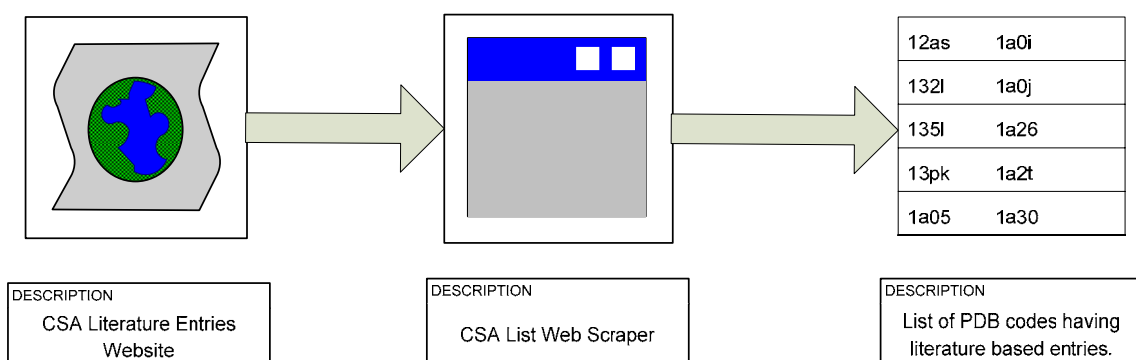
Note that this list includes only entries derived directly from the literature. There are many more entries inferred by sequence comparison.  
Click on a column heading to sort by that column.

PDB code	Name	EC number	CATH code
12as	Asparagine synthetase	6.03.01.0001	3.30.930.10
132l	Lysozyme	3.02.01.0017	1.10.530.10
135l	Lysozyme	3.02.01.0017	1.10.530.10
13pk	3-phosphoglycerate kinase	2.07.02.0003	3.40.50.1260 3.40.50.1270
1a05	3-isopropylmalate dehydrogenase	1.01.01.0085	3.40.718.10
1a0i	Dna ligase	6.05.01.0001	2.40.50.140 3.30.1490.70 3.30.470.30
1a0j	Trypsin	3.04.21.0004	2.40.10.10

**Figure 12.** List page of CSA literature based entries containing PDB notation with link to details page (orange ovals), name of an enzyme, EC number, and CATH code.

As can be seen from Figure 12, first column under heading “PDB code” provides a link to a page that contains the necessary catalytic site information. I have developed a Perl based Web Scraper to parse the web page in Figure 3 and produce a list of PDB codes for which literature entries are available in CSA. Source code for this software is available in Appendix B Part I. It takes advantage of **HTML::TableExtract** Perl module obtainable from CPAN

(<http://search.cpan.org/dist/HTML-TableExtract/>). This module is designed for extracting the content contained in tables within an HTML document, either as text or encoded element trees. A more detailed explanation of HTML::TableExtract capabilities can be seen at the following URL: <http://www.mojotoad.com/sisk/projects/HTML-TableExtract/tables.html> . A simplified Figure 13 outlines the process that starts from saved webpage in Figure 12 that is passed as a command line parameter to CSA List Web Scraper.



**Figure 13.** Schematic diagram of CSA Literature based entries list Web Scraper process flow.

The next step is to follow each individual link in the first column of Figure 3 and web scrape the necessary information on every webpage. Figure 14 shows typical page that user would encounter after following that link. Colored ovals mark the information of interest that either needs to be extracted or taken into consideration during the process. A help page that

discusses all information found on the page is located at [http://www.ebi.ac.uk/thornton-srv/databases/CSA/csa\\_eg.html#CSAentriesLinks](http://www.ebi.ac.uk/thornton-srv/databases/CSA/csa_eg.html#CSAentriesLinks).

**EMBL-EBI**  
European Bioinformatics Institute

Get Nucleotide sequences

EBI Home About EBI Groups Services Toolbox Databases Downloads Submissions

Catalytic Site Atlas Version 2.1.9

Find Annotated Site: PDB code:  Search Swiss-Prot code:  Search EC number:    Search

### CSA entry for 12as Original Entry

Title:	Ligase		
Compound:	Asparagine synthetase		
Mutant:	Yes		
UniProt/Swiss-Prot:	P00963-ASNA_ECOLI	EC Class:	6.3.1.1
Other CSA Entries:	Homologues of 12as Entries for UniProt/Swiss-Prot: P00963 Entries for EC: 6.3.1.1	Other Databases:	PDB entry: 12as PDBsum entry: 12as UniProt/Swiss-Prot: P00963 IntEnz entry: 6.3.1.1 KEGG entry: 6.3.1.1 MACiE mechanism: 12as EzCatDB entry: S00413

### Literature Report:

Mechanism: A two step reaction is postulated in which the amino acid is activated by ATP forming an aminoacyl-adenylate intermediate, to which an ammonia molecule is added.

Sites:

Catalytic Site (Get help with this section)

Found by Literature reference (Structural analysis and templates exist for the 12as family)

Residue	Chain	Number	UniProt number	Functional part
ASP	A	46	46	Sidechain
ARG	A	100	100	Sidechain
GLN	A	116	116	Sidechain

Catalytic Site (Get help with this section)

Found by Literature reference (Structural analysis and templates exist for the 12as family)

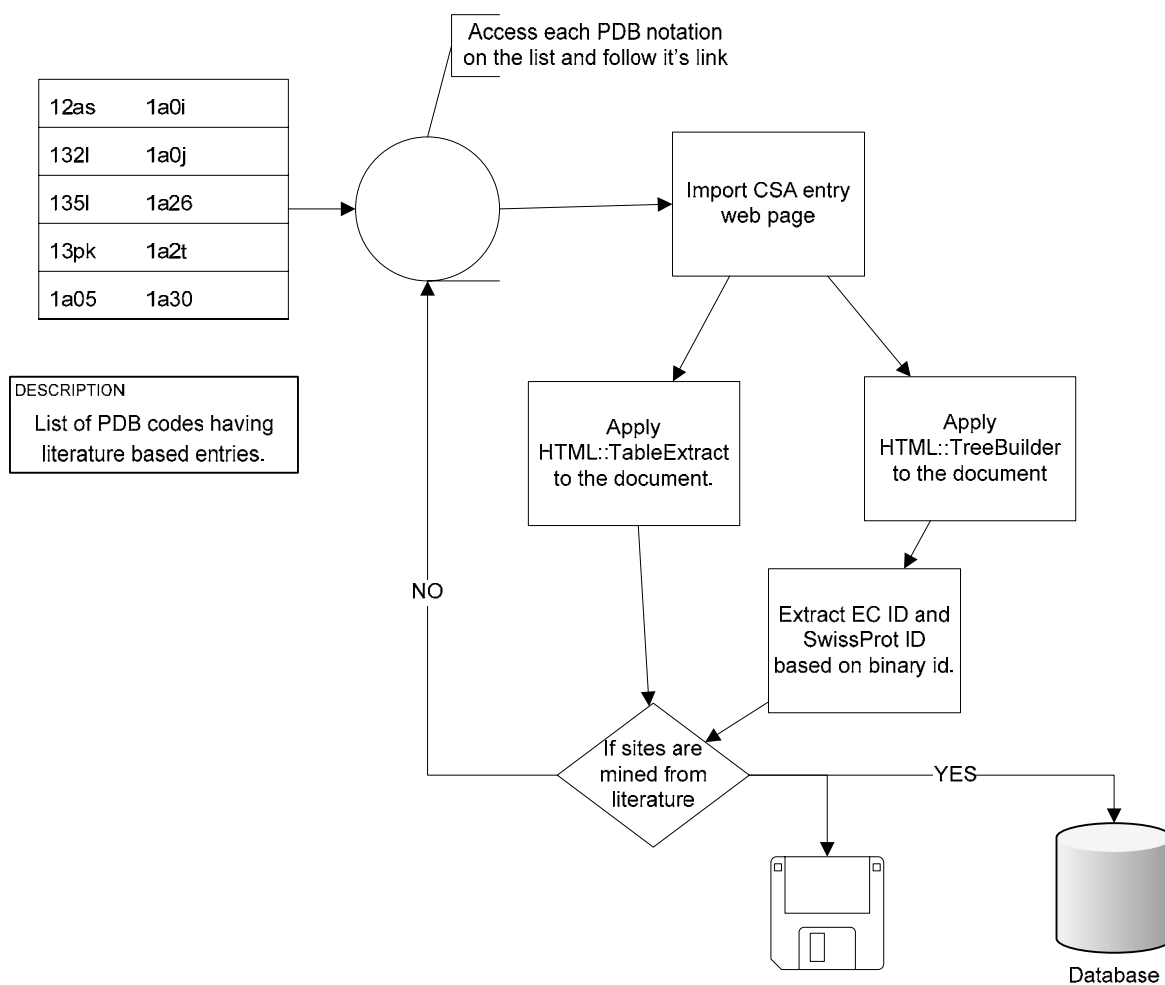
Residue	Chain	Number	UniProt number	Functional part
ASP	B	46	46	Sidechain
ARG	B	100	100	Sidechain
GLN	B	116	116	Sidechain

Use the check-boxes to select site(s) to view on the 3D structure in RasMol, and press

**Figure 14.** Typical CSA entry detail page with literature catalytic sites information. Data need for pipeline is marked by orange ovals.

After exhaustive trials it was determined that the best solution is to split each page into a customized Binary Search Tree (BST) of elements using **HTML::TreeBuilder** to avoid irregularities due to missing information and differences in representation from one CSA detail page to another. HTML::TreeBuilder assigns a binary tree node ID to all elements on the web page. This allows identification of relative distance between the objects on the page such as HTML tables or text entries. Also, it allows a way around missing or inconsistent information from page to page by providing alternative root for data search.

Figure 15 outlines a process of Web Scraping a typical CSA entry detail page. The output produced by this software is stored both in the GO Mappings database and in individual files for each entity. The database solution is envisioned for the second revision of the pipeline. Source code for the CSA entity Web Scraper is located in Appendix B Part II.



**Figure 15.** Flow chart of logic in CSA & Swiss-Prot catalytic site entry page Web Scraper.

Similar procedure is used to obtain active site profiles from Swiss-Prot database. However, the information there is much more organized and the use of HTML::TreeBuilder is not required. Active site profiles from CSA and Swiss-Prot are combined to supplement each other.

## ***INCORPORATING FDS & ACTIVE SITE INFORMATION TO SUPPLEMENT EPD***

Once all of the components of the pipeline have been gathered (EPD predictions, FDS, and active sites) it is now necessary to combine the subunits into whole in order to complete creation of the pipeline and commence verification of the initial hypotheses. The verification is a two-fold process because EPD produced a broader 3-digit EC (e.g. 1.1.1) function predictions as well as more specific 4-digit (e.g. 1.1.1.49) ones.

### **3-digit EC Predictions**

In the case of 3-digit EC predictions, active site positions are estimated during the subfamily creation through sequence homology. It should be emphasized that estimated positions represent data-based FDS, and not active sites, similar to approach used in EFICAz. Figure 16 displays a typical FDS content for 3-digit EC family.

```

FAMILY>  DHB7_HUMAN

ACC> 1.000000
SEN> 0.791667
SPEC> 1.000000

  pos res  freq  sc-t  sc-ho  sc-he
POS>  91 T  1.000  2.340  1.822  1.732
POS> 173 N  1.000  1.248  1.822  0.378
POS> 172 N  0.958  1.090  1.520  0.901
POS> 225 N  0.958  0.838  1.520  0.403
POS> 276 Y  0.958  0.720  1.520  0.044
POS> 293 E  0.875  0.683  1.027  0.987
POS> 293 D
POS> 255 S  0.958  0.610  1.520 -0.473
POS> 252 N  0.875  0.436  1.166  0.277

```

**Figure 16.** FDS Profile for DHB7\_HUMAN subfamily featuring accuracy, sensitivity, specificity parameters for the profile and for each active site position.

These FDS are stored in individual files having a filename consisting of “<EC number>.<family name>.fis”. For example, profile in Figure 16 would be stored in the file named 1.1.1.DHB7\_HUMAN.fis. Each file contains fairly large amount of information that can potentially be used in the analysis of accuracy of function prediction. Included in the file are: accuracy, sensitivity, and specificity of each FDS set in discriminating the enzymatic function corresponding to given FDS from other enzymatic and non-enzymatic functions on the training set, as well as frequency of each position in the family MSA calculated to be of functional importance.

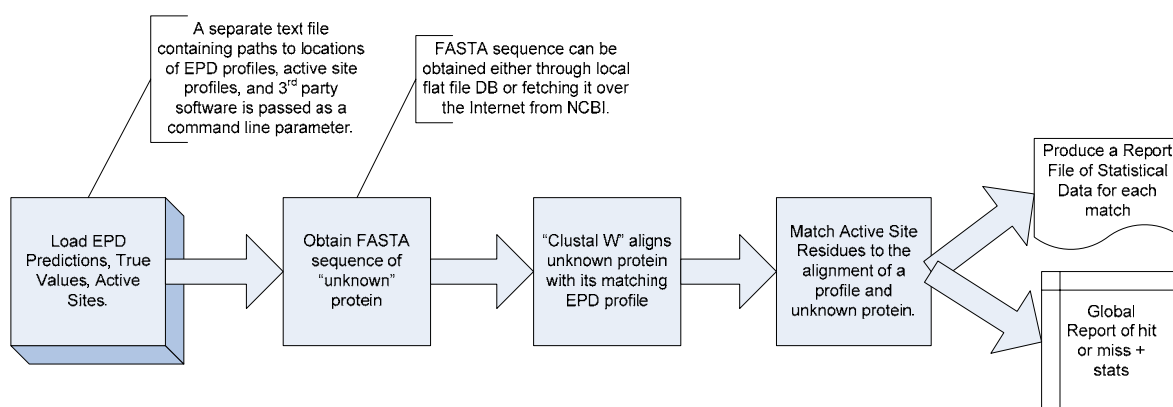
```

>>>>
3BHS_CANFA + 0 1.1.1 3BHS1_MESAU 207 1746
3BHS_CANFA + 4577 5.3.3 3BHS1_MOUSE 146 1625
3BHS_CANFA + 4578 5.3.3 3BHS_UACCA 152 1595
3BHS_CANFA + 30 1.1.1 DFRA_PETHY 312 1264
3BHS_CANFA + 17 1.1.1 ALD2_SPOSA 315 1261
3BHS_CANFA + 29 1.1.1 DFRA_ARATH 315 1261
3BHS_CANFA + 4593 5.3.3 TYRP2_HUMAN 964 1041
3BHS_CANFA + 62 1.1.1 FCL_CRIGR 231 609
3BHS_CANFA + 145 1.1.1 RFBD_RHISN 152 260
3BHS_CANFA + 784 1.6.5 NUEM_BOVIN 72 75
3BHS_CANFA + 1992 2.7.1 M3K4_MOUSE 60 65

```

**Figure 17.** Several predictions of an unknown protein named on the left to several EC numbers and subfamilies in the middle. Each subfamily threshold is outlined in red with the raw score next to it as the last column.

An example of EPD predictions format is outlined in Figure 17. Each unknown “3BHS\_CANFA”, repeated in the left column is followed by the `+` indicating a positive hit. Next column features a unique subfamily identification number followed by the EC number predictions and subfamily name. Last two columns contain subfamily unique threshold and a raw score of the unknown matched against the subfamily. All of that information is parsed by the pipeline and unknown protein is aligned with each of its matches using ClustalW. After alignment complete FDR positions are analyzed to determine if conservation has occurred. The results and statistical information are recorded in two report files: individual file for each match and global report for all available unknowns. Figure 18 depicts general flow of logic in the pipeline.



**Figure 18.** Flow chart diagram outlining major subunits and direction of information passage in "EPD + Functional Sites" enzyme function prediction pipeline.

Appendix F contains an example of a complete report for one unknown that has been matched to three EC numbers and several profiles. Results are summarized at the bottom of the report and duplicated in the Global Report file. Source code for 3-digit pipeline is located in Appendix D and takes advantage of Perl module CDD from Appendix C that contains individual subroutines of a pipeline.

### 4-digit EC Predictions

Predictions for more precise 4-digit EC numbers are done exactly the same as 3-digit ones with one notable exception. FDR for 4-digit EC numbers are derived from outside source, either CSA or Swiss-Prot literature derived manually curated entries. Therefore, it is necessary to insert a step where positions recorded in the literature are translated into

positions obtained after ClustalW multiple alignment of a profile with matching unknown. Otherwise, Figure 18 is sufficient to describe 4-digit prediction as well. Source code for 4-digit pipeline is located in Appendix E and also takes advantage of Perl module CDD from Appendix C that contains individual subroutines of a pipeline.

## ***IMPLEMENTATION***

### **Software Package**

Pipeline software operates from a command line prompt of either UNIX or Windows operating systems. It assumes that RPS-BLAST, ClustalW, and BioPerl, are already installed and functioning correctly on the system. For UNIX operating systems software is packaged as a collection of separate files: one for each of the 3- and 4-digit prediction types and common subroutine module. For Windows operating system, separate files and common subroutine module are packaged together into a single executable file. Software module accepts one command line parameter in a form of a text file that contains paths to various databases, EC number families' locations, and output directories.

### **High-Performance Computing Details**

Several validation runs of the pipeline were initiated on different platforms to calculate execution times:

1. Intel Pentium 4 3.0 GHz 1MB L2 800 MHz front-side bus CPU, 1.00 GB of RAM, MS Windows XP Professional @ home.
2. 4 processor Intel Xeon 2.8 GHz 2x1MB L2 800 MHz front-side bus CPU, 2.00 GB of RAM, MS Windows Server 2003 @ BHSAL.

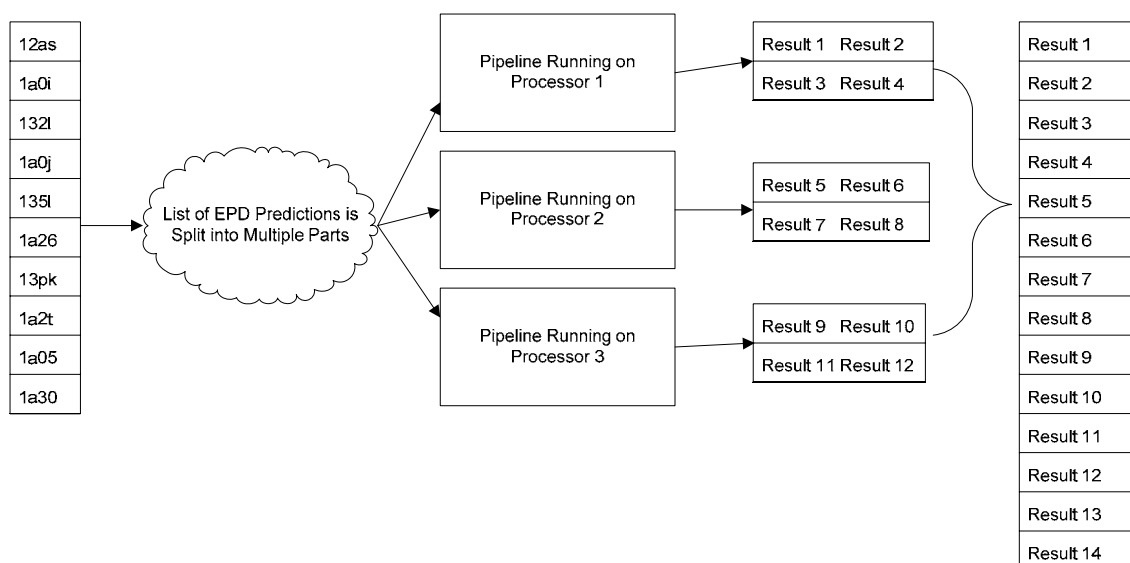
3. 8 processor 64-bit Sun UltraSparc IIe 500MHz 256 KB L2 CPU, 2.00 GB of RAM, Sun Solaris 9 @ ExonHit Therapeutics, Inc.

The computing platforms discussed here are used in prototyping and do not reflect the computing environment currently being used for the function prediction at BHSAL.

### **Serial & Parallel Execution Time**

Execution was tested on a set of previously generated predictions for ~10,000 enzymes selected as a validation set and discussed in the next section. Approximate serial execution time on platform 1 described in a previous section varies from 72 to 96 hours. Platforms 2 & 3 do not offer any advantages due to approximately equal throughput which depends mostly on processor clock speed.

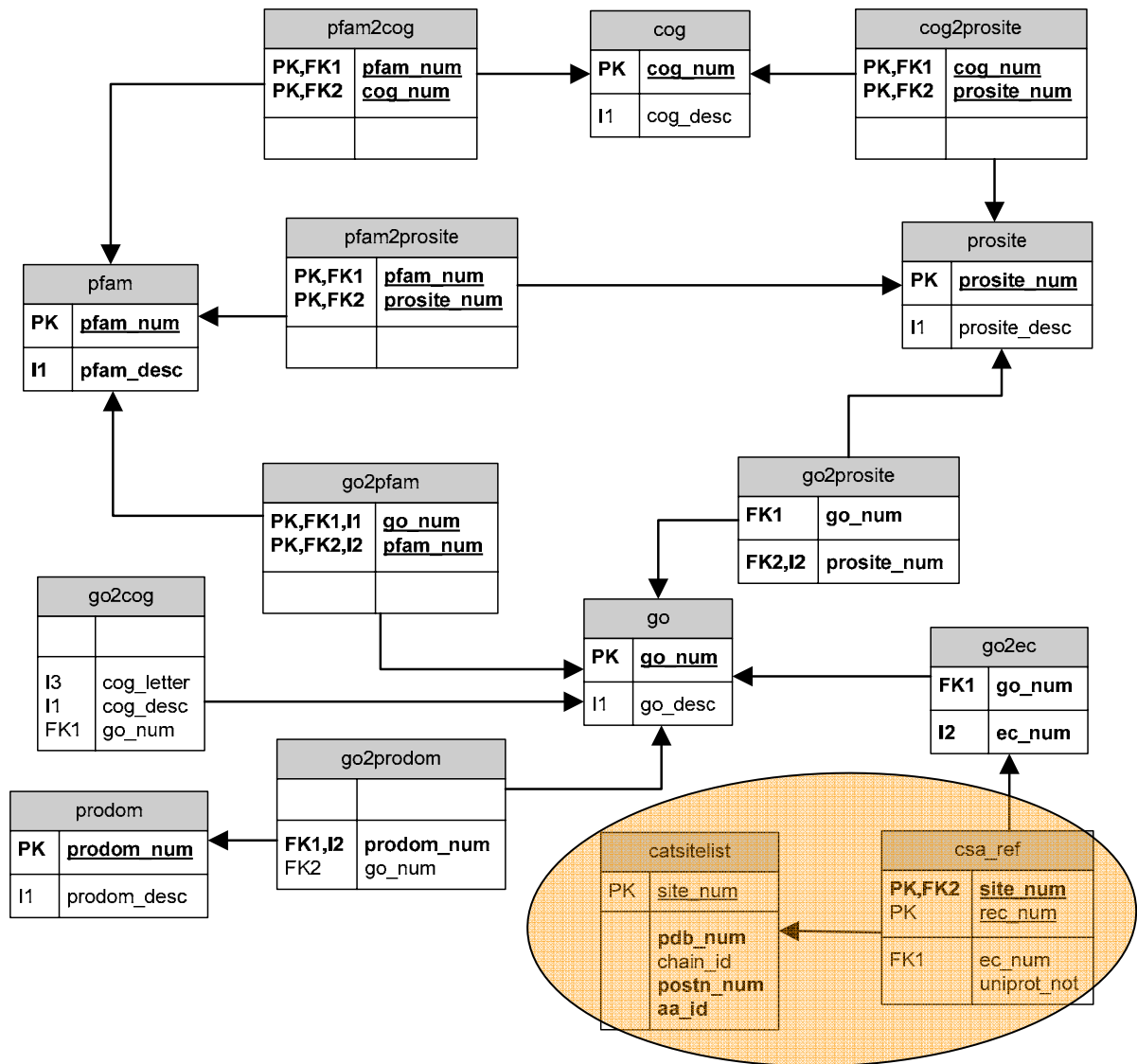
Parallel execution, on the other hand, offers a very substantial increase in performance on platforms 2 & 3. Parallel implementation of the pipeline involves dividing the list of original EPD predictions into several sub lists of however many parallel processes one wishes to run concurrently. Running a pipeline on 4 processors shortens total execution time down to 24 hours. A more efficient approach, not used here, would be to estimate FDS conservation for a group of unknown proteins immediately after EPD prediction. Figure 19 visualizes a concept of parallel execution of a prediction pipeline on several processors of a single platform or several different platforms.



**Figure 19.** Flow chart diagram outlining a concept of parallel execution of a pipeline.

## Generated Database

Initial design and implementation were done on a MySQL 5.0 RDBMS platform (<http://www.mysql.org>) using SQLyog 5.02 Database Manager ([http://www.webyog.com/sqlযোগ/index\\_sqlযোগfree.php](http://www.webyog.com/sqlযোগ/index_sqlযোগfree.php)). Figure 20 contains final ER diagram for the database generated in the course of this research. The difference between Figure 10 & Figure 20 is highlighted in orange on the latter figure and consists of literature Active Sites Information obtained from CSA. Details of the GO Mappings database tables are found in Appendix G. Future version of this pipeline would rely solely on RDBMS for storage and transfer of information between components of the pipeline.



**Figure 20.** Final database schema containing FDS information highlighted in orange.

## ***ENZYME VALIDATION SET***

In order to validate pipeline prediction ability a control set of 13,239 enzymatic proteins with independently assigned function data was selected. These enzymes were split off from the pool of sequences downloaded from ENZYME database specifically to act as “unknowns” during validation experiments. Remaining sequences were used in generation of subfamilies and corresponding profiles. Thus, by inputting into a pipeline a never before encountered sequence with known EC number and then assessing whether it was associated with the correct 3- or 4-digit EC number family, accuracy of prediction can be assessed.

## ***EVALUATION PARAMETERS***

Pipeline prediction performance was evaluated using two quantities: accuracy and coverage.

### **Accuracy**

Accuracy, in this context, is defined as a fraction of True Positives (TP) obtained from all predictions. It is calculated by equation in Figure 21.

$$Accuracy = \frac{TP}{TP + FP}$$

**Figure 21.** Equation for calculating accuracy based on the number of correct predictions (TP).

TP is defined as a prediction that corresponded to unknown's EC number designation as obtained from ENZYME database. FP is an incorrect prediction of unknown's EC number.

## Coverage

Coverage is a fraction of unknown input proteins for which EPD + FDR, which was tested here, was able to make a prediction. Coverage is greatly influenced by thresholds and filters that are applied to remove FP predictions. Some unknowns have only a single low score prediction that gets rejected by stringent filtering. This situation, in effect, leaves an input unknown without a prediction, thus reducing the coverage quantity. The equation for computing coverage is outlined in Figure 22.

$$\textit{Coverage} = \frac{\text{Number Proteins with Predicted Function}}{\text{Total Number of Tested Proteins}}$$

**Figure 22.** Equation for calculating input coverage based on fraction of input proteins with acceptable predictions.

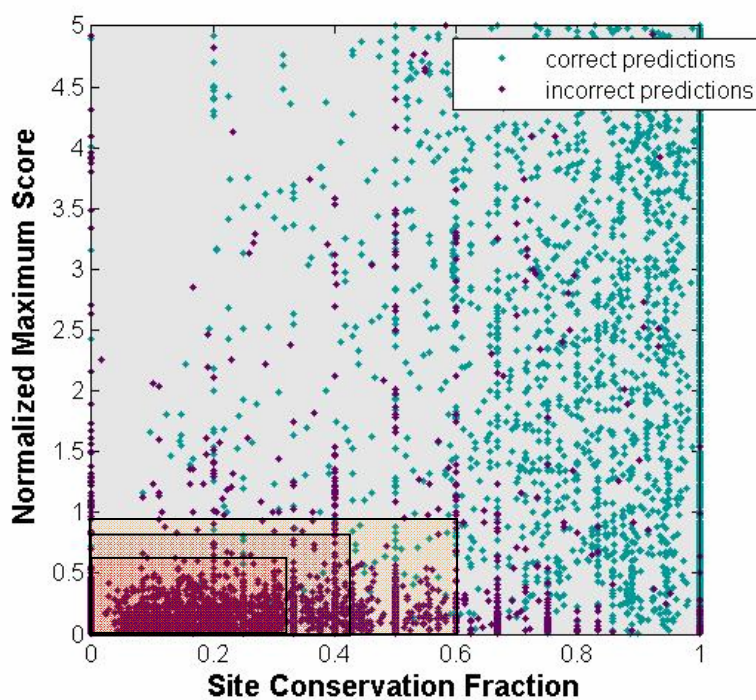
## Results & Discussion

### *3-DIGIT EC FAMILIES*

The predictions for 3-digit families had the following distributions. Out of input of 13,239 unknown proteins of the validation set, 12,749 were assigned a prediction by EPD. Following a formula outlined in Figure 22 coverage of 96.3% was obtained. The total number of all predictions was 20,211. This number is higher than the number of input proteins because some unknowns have been predicted, and correctly so, to belong to two or more EC families. For example, 3BHS\_CANFA in Figure 17 is actually a member of 5.3.3.1 as well as 1.1.1.145 according to Swiss-Prot database.

Out of 20,211 predictions, 12,874 were assigned correctly (True Positives) and 7,337 were predicted incorrectly (False Positives). The two values of interest are the Normalized Maximum Score (NMS) and Site Conservation Fraction (SCF). First value is computed by normalizing the scores produced by EPD (Figure 5) and taking a maximum match score for each unknown tested. Second value is calculated by dividing the number of matching FDS/Active Sites by the total number that is available for this unknown. Together these values can serve as a reliable indicator of whether prediction was correct. Figure 23 contains a snippet (NMS = 0 to 5.0 and SCF = 0 to 1.0) of a most densely populated area of the

graph of True Positive (TP) and False Positive (FP) values for 3-digit EC predictions. Calculated accuracy for unfiltered 3-digit predictions is 63.7%.



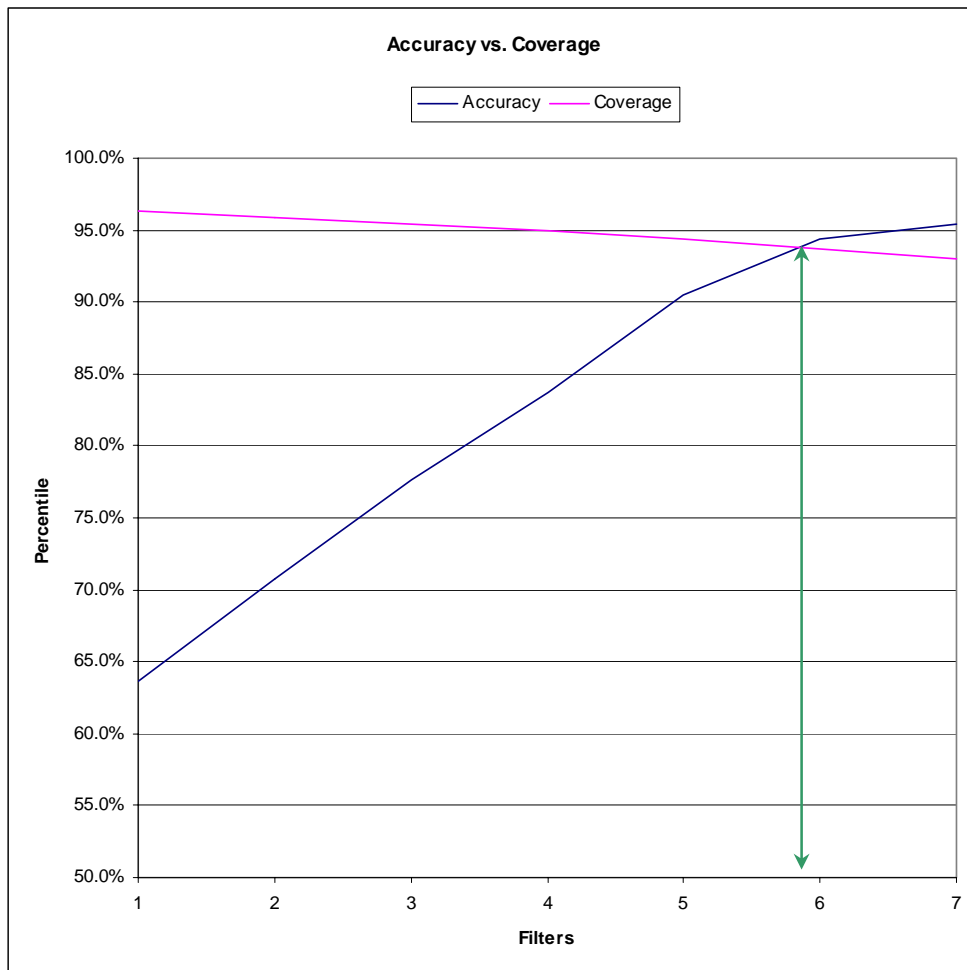
**Figure 23.** NMS vs. SCF plot. Distinct clustering of TP (turquoise) and FP (burgundy) predictions in different regions of the plot allows application of filtering (colored rectangles) to remove FP and increase accuracy.

Colored rectangles indicate areas that would be filtered out when different threshold values of NMS and SCF would be applied to eliminate FP. Predictions inside that area would be rejected, possibly resulting in decrease of coverage. Threshold values can be identified on NMS and SCF axis that would create a rectangular boundary that separates area dominated by TP from the area of FP clustering. By visual estimation of Figure 23, it is evident that

vast majority of FP predictions is contained in the lower left quadrant of the plot. FPs appear to have NMS of less than 1.0 and SCF of less than 0.8. To find optimum threshold values for NMS and SCF at which majority of FPs is rejected without significant loss of coverage their values a gradually increased from 0.1 (see Table 1) and values of accuracy and coverage are plotted on Figure 24. The point where Accuracy plot intersects Coverage on Figure 24 represents optimal percentile, around 94%. In some cases, obtaining high accuracy might be desirable at the expense of coverage and more stringent filtering criteria could be used.

**Table 1.** Effect of increase in stringency of filters on accuracy and coverage of 3-digit family predictions. Optimum balance between accuracy and coverage is highlighted in yellow (~94%).

TP	FP	Accuracy	Coverage	Filter
12,874	7,337	63.7%	96.3%	NONE
12,855	5,298	70.8%	95.9%	NMS>0.1 + SCF>0.1
12,842	3,712	77.6%	95.4%	NMS>0.2 + SCF>0.2
12,818	2,491	83.7%	95.0%	NMS>0.3 + SCF>0.3
12,787	1,341	90.5%	94.4%	NMS>0.5 + SCF>0.4
12,706	753	94.4%	93.7%	NMS>0.7 + SCF>0.5
12,618	612	95.4%	93.0%	NMS>0.9 + SCF>0.6

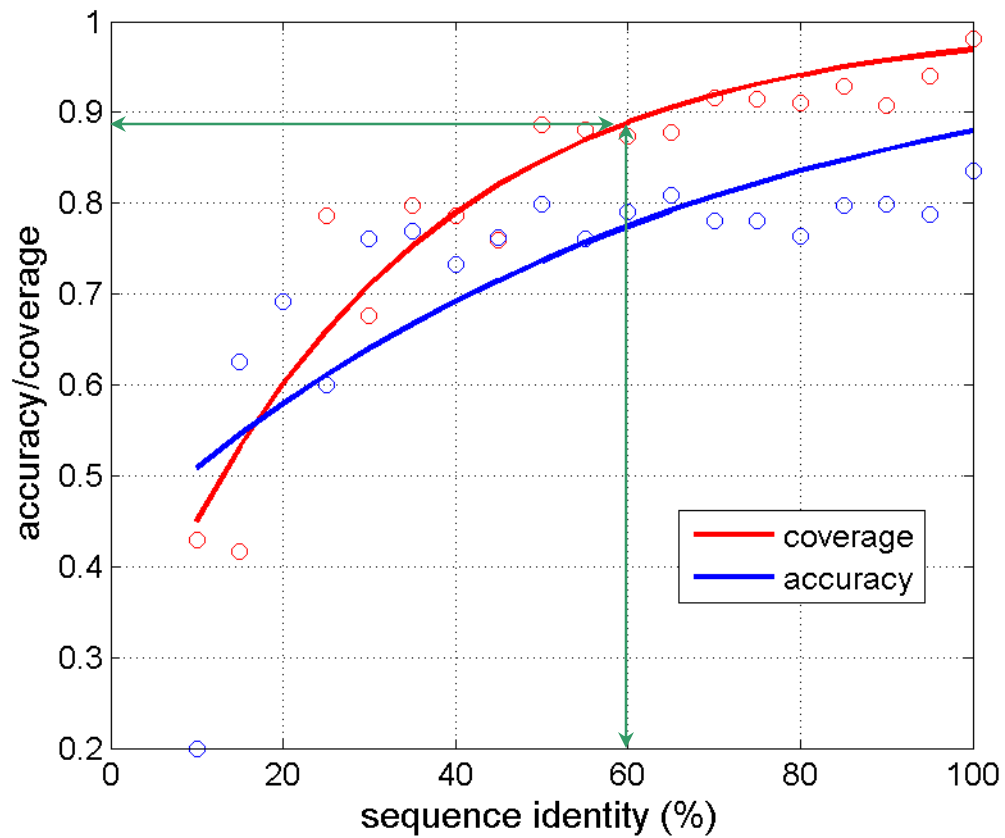


**Figure 24.** Accuracy (blue) vs. Coverage (pink) plot indicating that optimum trade off between them occurs during application of filter #6 (green double arrow) when values having  $NMS < 0.7$  and  $SCF < 0.5$  are rejected.

## ***DISCUSSION OF 3-DIGIT RESULTS***

Trade-off between accuracy and coverage is presented in **Figure 25** below as a function of sequence identity for the EPD prediction without FDS filtering. Although FDS would help in accuracy increase, reduction of coverage may not be acceptable.

Current estimates indicate that at 60% sequence identity, accuracy approaches 80% and coverage approaches 90%. These numbers increase further as more sequence identity is achieved between a profile and an unknown protein.



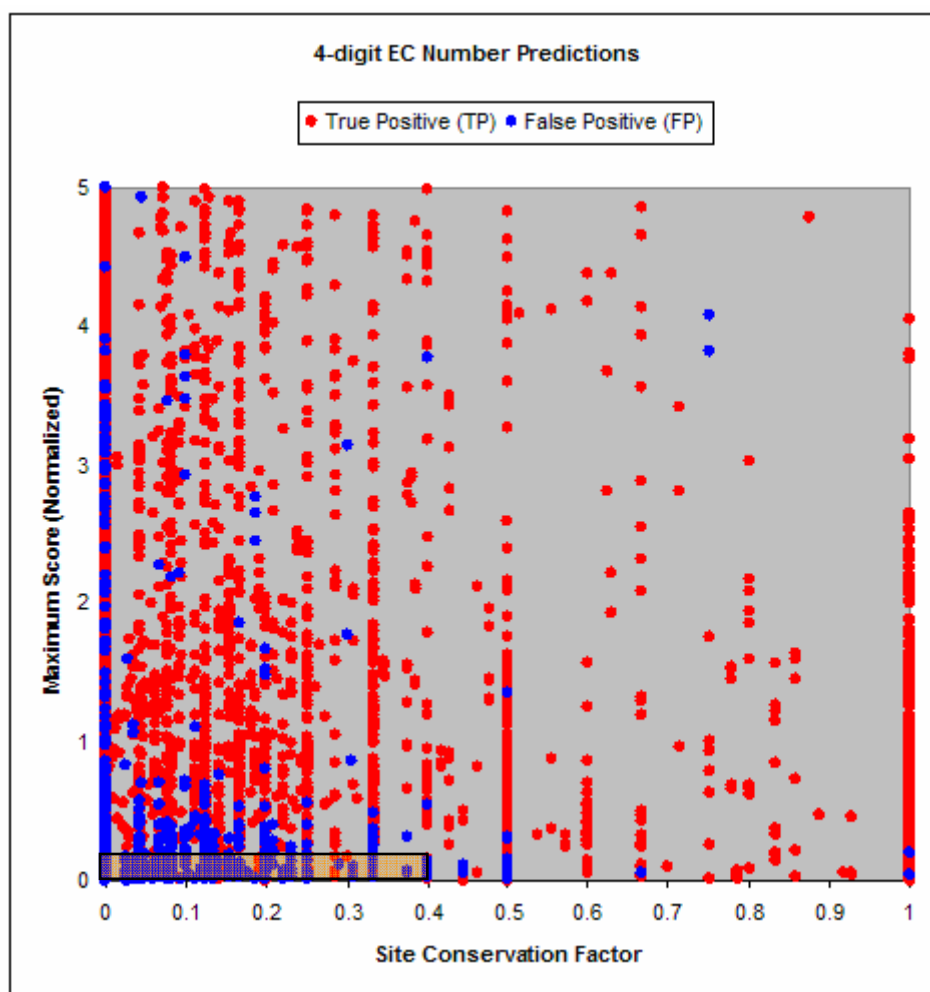
**Figure 25.** Plot of Accuracy & Coverage vs. Sequence Identity indicating a direct relationship between these values. At 60% sequence identity accuracy of prediction is just under 90% (shown by green double arrows).

## ***4-DIGIT EC FAMILIES***

Accuracy and Coverage for 4-digits are calculated in the same manner as for 3-digit EC number predictions. The difference with 4-digits consists in greater variety of data filtering. Lack of available literature-mined Active Sites and/or suitable EPD profiles allows easy discrimination of data.

Input consisted of the same set of proteins as in 3-digit predictions. Total number of predictions for 4-digit EC by EPD was 13,393 and included multiple predictions for many input sequences. This set is represented in **Figure 26**. Coverage was calculated to be 80.4%. Out of that number 10,572 were TP, and 2,821 were FP. Accuracy was calculated to be 78.9%. Results for 4-digit predictions required several levels of manipulation and filtering. Predictions that had zero threshold ( $T_c=0$  and implies that profile could not be generated) were eliminated from the set. That narrowed the set down to 11,969 total predictions with 10,504 TP and 1,465 FP for 79.1% of total coverage. Accuracy has increased to 87.8%. Further filtering focuses on exclusion of data in lower left portion of the graph. Shaded region in **Figure 26** represents data that was filtered out to improve the accuracy using the specified filter thresholds. Accuracy and coverage values are summarized in **Table 2**. **Figure 27** contains a plot of Accuracy vs. Coverage and provides visual estimate of optimal trade-

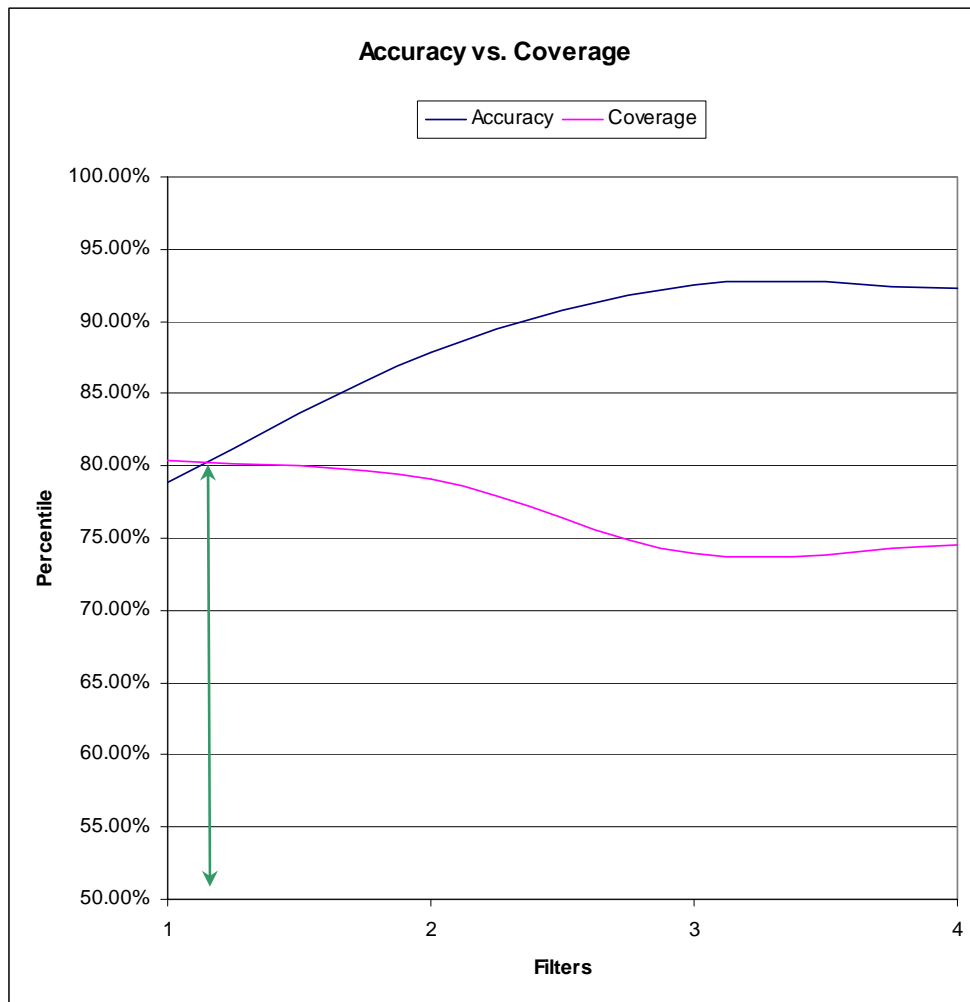
off between those two values which appears to be around 80th percentile and corresponds to no application of any filters.



**Figure 26.** NMS vs. SCF for 4-digit EC family predictions indicating weak clustering of TP and FP predictions. Orange rectangle indicates the most stringent filter applied to predictions.

**Table 2.** Effect of increase in stringency of filters on accuracy and coverage of 4-digit family predictions. Optimum balance between accuracy and coverage is highlighted in yellow. Increase by 9% in accuracy is achieved by 1% decrease in coverage.

<b>TP</b>	<b>FP</b>	<b>Accuracy</b>	<b>Coverage</b>	<b>Filter</b>
10,572	2,821	78.9%	80.4%	NONE
10,504	1,465	87.8%	79.1%	EPD>0
9,860	796	92.5%	73.9%	EPD>0 * NSM>0.1
9,953	830	92.3%	74.5%	EPD>0 * (NMS>0.1 + SCF>0.4)



**Figure 27.** Accuracy vs. Coverage plot of 4-digit families predictions. Optimum trade off between Accuracy and Coverage occurs without application of filters.

## ***DISCUSSION OF 4-DIGIT RESULTS***

In general, accurate estimation of 4-digit EC numbers from sequences is difficult, since it involves identification of the substrate specificity, which could be very ambiguous. Also, experimental and literature information for active sites is very scarce. Often, reference active sites are not conserved in the homology-based EPD families and the query sequence has a high probability to be incorrectly aligned at the critical positions unlike the situation with 3-digit EC numbers, where FDS are determined using given EPD alignments. Such approach was not pursued for 4-digit EC numbers due to the reduced size of subfamilies for most EC numbers, which would lead to incorrect position conservation estimation. Additional check of the active site profiles has to be executed to verify that active sites from literature entries correspond correctly to the profiles compiled by EPD. The average fraction of conserved active site was selected as a measure of conservation to alleviate the problems caused by incorrect alignments; however, average conservation could be very small due to a large number of subfamilies for some 4-digit families, as well as occurrences of multiple redundant active sites in Swiss-Prot. Improved methods to use reference active site in EPD need to be found.

Elimination of false positive predictions based on the average fraction has a small effect on accuracy. On the other hand, analysis of the testing results indicated that a high average

conservation fraction can be used as evidence for prediction reliability. For example, out of 514 predictions with average conservation fraction larger than 0.5, only six predictions are incorrect, resulting in reliability of 99%.

Thus we can tentatively define two qualitative regions of reliability: (a) high, when prediction has the NMS larger than 1 and the SCF  $> 0.5$ ; and (b) low for all other admissible cases.

We eliminate as unreliable all predictions that have average conservation fraction less than 0.4 and NMS  $< 0.5$ .

## ***CONCLUSIONS & FUTURE WORK***

The problem of prediction of function from amino-acid sequence is far from being satisfactory solved. Numerous pipelines have been built to transfer annotation from known proteins to the large collection of sequences derived from genomic projects. Many of the methods that have been applied to function prediction work part of the time but none is perfect. Therefore, the most sensible strategy is to subject the target to a battery of different prediction methods. Should several methods concur, there is greater confidence in their prediction.

Results obtained in both 3-digit and 4-digit EC number predictions indicate that implemented pipeline can provide high degree of accuracy, especially when fine-tuned filters are applied. However, increase in accuracy comes at the expense of the coverage. An addition of another dimension in the form of FDS from homology (3-digits) or CSA literature entries (4-digits) has varying effect on accuracy. If FDS are available for every subfamily, as with 3-digit families, it significantly increases the accuracy. Similarly, if FDS are sparse, as with 4-digit predictions, an increase in accuracy is not evident. However, my analysis indicates that it is possible to obtain modest improvement in coverage when a combination of NMS and SCF thresholds is used rather than a simple threshold of NMS.

Validation experiments performed for these study yielded increase in accuracy of greater than 30% for 3-digit EC number predictions while coverage decreased only by slightly more than 3%. It was noted that both accuracy and coverage greatly depend on sequence similarity between a query and corresponding match. Gains in accuracy of 4-digit predictions were promising at 13.5%, while loses in coverage reached 6.5%, twice that of 3-digit.

As more and more methods and data become available, housed on separate servers or packaged in different applications, so it becomes more difficult to gather and combine the necessary information for incorporation into pipelines. This study attempted to bridge numerous external protein annotation databases by creating a cross-reference database using the most general classification of function that is produced by the Gene Ontology Consortium. In addition, cross-reference database was expanded to incorporate information about enzyme features such as active sites, FDR, etc.

Future work will concentrate on incorporating additional functionality into the pipeline. Some of the proposed directions include combination with InterPro. Incorporating such a vast repository of protein data into current pipeline would add another dimension to selecting a correct prediction for unknown sequence. One of the ways such integration can be accomplished is by joining “Ontology Mappings” with InterPro using Pfam, COG, or other notations as common reference point.

Combining enzyme prediction with pathway information is another direction that will be explored in the future. By combining related pathways and analyzing the structure and homology of the enzymes that catalyze them, predictions for the unknown proteins can be done more intelligently using a higher level of information.

It is expected that software developed in this project will find applicability in genome-wide protein function recognition, leading to accelerated design of advanced drug and vaccine products for the DoD Force Health Protection mission.

## **Disclaimer**

The opinions and assertions contained herein are the private views of the author and are not to be construed as official or as reflecting the views of the Department of Defense Biotechnology High Performance Computing Software Application Institute.

## Appendix A

```
#!/usr/bin/perl -w
#
# File: ParseGoMaps.pl
# Name: Seth Johnson
# Date: 09/23/05
#
#
use strict;
use warnings;

# Open the file given as the first argument on the command line
my $filename = $ARGV[0];

# perform actual parsing
read_db2go_file($filename);

exit;

#####
# Open the input file, determine it's type, and parse the information
# into output files.
#####
sub read_db2go_file
{
    my $line;
    my (@words, @cogs);
    my ($filename) = @_;
    open(INFILE, $filename) or die "Can't open $filename";
    open(OUTFILE, ">map2go.csv") or die "Can't open map2go.csv: $!";
    open(OUTText, ">extrnl.csv") or die "Can't open extrnl.csv: $!";
    open(OUTgo, ">go.csv") or die "Can't open go.csv: $!";

    while ($line = <INFILE>)
    {
        $/ = "\n";
        chomp $line;
        if ($line =~ /^Pfam:/) # line starts with a "Pfam:"
        {
            @words = split(/(?:Pfam:)(\w+)\s+(.*?)\s*(.*?)\s\;.*:(\d+)/, $line);
            print OUTFILE $words[4].",".$words[1]."\n"; # save line to an output file
            print OUTText $words[1].",".$words[2]."\n";
            print OUTgo $words[4].",".$words[3]."\n";
        }
        elsif($line =~ /^COG:/) # line starts with a "COG:"
        {
            @words = split(/^COG:|\s:\sGO:|\s>\sGO:/, $line);
            if ($words[1] =~ /^w\s/)
            {
                @cogs = split(/(^.)\s/, $words[1]);
                $words[0]=$cogs[1];
                $words[1]=$cogs[2];
                for(my $ii = 3;$ii < scalar @words;$ii+=2)
                {
                    print OUTFILE $words[$ii]."\t".$words[0]."\t".$words[1]."\n";
                    print OUTgo $words[$ii]."\t".$words[$ii-1]."\n";
                }
            }
            elsif($words[(scalar @words)-1] =~ /^[^\.]/)
            {
                print OUTFILE $words[(scalar @words)-1]."\t".$words[1]."\n";
                print OUTgo $words[(scalar @words)-1]."\t".$words[(scalar @words)-2]."\n";
            }
            print OUTText $words[0]."\t".$words[1]."\n";
        }
        elsif($line =~ /^EC:/) # line starts with a "EC:"
        {
            @words = split(/^EC:|\s>\sGO:|\s:\sGO:/, $line);
            print OUTFILE $words[(scalar @words)-1]."\t".$words[1]."\n";
        }
    }
}
```

```

        #print OUTText;
        print OUTgo  $words[(scalar @words)-1]."\t".$words[(scalar @words)-2]."\n";
    }
    elseif($line =~ /^ProDom:/) # line starts with a "ProDom:"
    {
        @words = split(/(?:ProDom:)(\w+)\s+(.*?)\s>\sGO:(.*?)\s\;\sGO:(\d+)/, $line);
        print OUTFILE $words[4]."\t".$words[1]."\n";          # save line to an output file
        print OUTText $words[1]."\t".$words[2]."\n";
        print OUTgo $words[4]."\t".$words[3]."\n";
    }
    elseif($line =~ /^PROSITE:/) # line starts with a "PROSITE:"
    {
        @words = split(/(?:PROSITE:)(\w+)\s+(.*?)\s>\sGO:(.*?)\s\;\sGO:(\d+)/, $line);
        print OUTFILE $words[4]."\t".$words[1]."\n";          # save line to an output file
        print OUTText $words[1]."\t".$words[2]."\n";
        print OUTgo $words[4]."\t".$words[3]."\n";
    }
    else #if($line =~ /^IPB/) # line starts with a "IPB:"
    {
        @words = split(/(\w+)\:\s(.*?)\t(\w+)\:\s(.*?)\t(\w+)\:(.*?)\t(\w+)\:\s\w+\:\s(.*)/, $line);
    }
}
close INFILE;
close OUTFILE;
close OUTText;
close OUTgo;
return;
}

```

## Appendix B Part I

```
#!/usr/bin/perl -w
use strict;

use HTML::TableExtract;
use File::Slurp;

my $filename = $ARGV[0];

my @pdb_lit;
#my $file_contents = read_file($filename);
my $te = HTML::TableExtract->new( headers => ["PDB code"] );
$te->parse_file($filename);

foreach my $ts ($te->tables) {
    #print "\nTable ('", join(',', $ts->coords), "):\n";
    foreach my $row ($ts->rows) {
        my $code_pdb = @$row[0]."\n";
        # $code_pdb =~ s/^\s+|\s+$//g;
        push(@pdb_lit,$code_pdb);
        #print "Length: ".length($code3);
    }
}
write_file( 'CSAlist.txt', @pdb_lit );
exit;
```

## Appendix B Part II

```

#!/usr/bin/perl -w
use strict;

use HTML::TreeBuilder;
use HTML::TableExtract;
use LWP::Simple;
use File::Slurp;
use DBI;

use CDD;

my $filename = $ARGV[0];
my @all_pdb = read_file($filename);

my $dbh = DBI->connect('dbi:mysql:gonap:localhost'.'.root'.'.perenos'.
    ( RaiseError => 1, AutoCommit => 1, mysql_auto_reconnect => 1 ));

my $ins_csa_ref = $dbh->prepare("insert into csa_ref (ec_num, uniprot_not, pdb_num) values (?,?,?)");
my $ins_catsitelist = $dbh->prepare("insert into catsitelist (pdb_num, chain_id, postn_num, aa_id) values (?,?.?.?.?)");

my @bad_code = qw/FSH SRH FGL SEG LLP NAD FAD CSE CSS NDP FHH CSD NAP SEC ASQ KCX TPQ HEM F4Q SCV CER EYS FEZ/;
my %is_bad = ();
for (@bad_code) { $is_bad{$_} = 1 }

foreach my $pdb_name (@all_pdb) {
    my ($pdb_id) = $pdb_name =~ m/^gn?$/;
    $pdb_id =~ s/^\/\s+|\s+\/$//g;
    print "PDB: ", $pdb_id;
    my $URL = "http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/CSA/CSA_Site_Wrapper.pl?pdb=".$pdb_id;
    my $content = get($URL);

    my $tree = HTML::TreeBuilder->new; # empty tree
    $tree->parse($content); # file($file_name);
    #print "Hey, here's a dump of the parse tree of $pdb_id:\n";
    #
    # $tree->dump; # a method we inherit from HTML::Element

    my $swiss_name = "";
    $swiss_name = $tree->address('0.1.1.4.1.0.2')->as_trimmed_text()
    if defined $tree->address('0.1.1.4.1.0.2');
    my $sec_number = "";
    unless (not ref($tree->address('0.1.1.4.3.0.0')))
    {
        $sec_number = $tree->address('0.1.1.4.3.0.0')->as_trimmed_text();
    }
    #if defined $tree->address('0.1.1.4.3.0.0');

    open(outCSA, ">".$sec_number.". ".$swiss_name.".csa") or die "Can't open Output File: $!";
    print outCSA "NAMES: ", $tree->address('0.1.1.4.1.0')->as_trimmed_text();
    print outCSA "\nEC_No: ", $sec_number;
    print "\nECs: ", $sec_number, "\n";
    $tree = $tree->delete;

    my $q_csa_ref = $ins_csa_ref->execute($sec_number,$swiss_name,$pdb_id) or die "Can't execute statement: $DBI::errstr";

    my $te = HTML::TableExtract->new( headers => [{"Literature reference"}] );
    $te->parse($content); # file($file_name);
    my ($dpth, $cnt);
    foreach my $tst1 ($te->tables) {
        #print "\nTable (" . join(',', $tst1->coords) . "):\n";
        $dpth = ($tst1->depth);
        $cnt = ($tst1->count);
        $dpth++;
        $cnt--;
    }
    my $tdata = HTML::TableExtract->new(depth => $dpth, count => $cnt, headers => [{"Residue", "Chain", "Number", "UniProt number"}]);
    $tdata->parse($content);

    # Examine all matching tables
    foreach my $ts ($tdata->tables) {
        #print "\nTable (" . join(',', $ts->coords) . "):\n";
        print outCSA "\n";
        foreach my $row ($ts->rows) {
            my $code3 = $row[0];
            $code3 =~ s/^\/\s+|\s+\/$//g;
            #print "Length: ", length($code3);
            my $nrc_num = $row[2];
            $nrc_num =~ s/^\/\s+|\s+\/$//g;
            if (length($code3) == 3 && not exists($is_bad{$code3}) )
            {
                if ($nrc_num eq "0") { $nrc_num = $row[2]; }
                my $one_lett_aa = CDD::HTML::adIn1($code3);
                print outCSA "\nPOS\t"; join("\t", $nrc_num, $one_lett_aa);
                my $q_catsitelist = $ins_catsitelist->execute($pdb_id,$row[1],$nrc_num,$one_lett_aa) or die "Can't execute statement: $DBI::errstr";
            }
        }
    }
}

close outCSA;
$ins_csa_ref->finish;
$ins_catsitelist->finish;
$dbh->disconnect;
}
exit;

```

## Appendix C

```

package CDD;

use strict;
use warnings;

use Bio::Tools::Run::StandAloneBlast;
use Bio::SeqIO;
use Bio::SearchIO;
use Bio::DB::Flat;
use Bio::AlignIO;
use Bio::Align::AlignI;
use Switch;
use DBI;
use HTML::Parser ();
use LWP::Simple;
use File::Slurp;

sub CDD_rpsBLAST
{
    my($seq_file,$e_value,$out_file) = @_;

    my $factory = Bio::Tools::Run::StandAloneBlast->new(db => 'CDD',
                                                       expect => $e_value);

    # There's one peptide per file assumed
    my $strng = Bio::SeqIO->new(-file=> $seq_file, -format => 'Fasta');
    my $input = $strng->next_seq();

    $factory->F('T');
    $factory->m('7');
    $factory->outfile($out_file);
    my $blast_report = $factory->rpsblast($input);
}

sub CDD_XMLblastRprtReader
{
    my($xml_file, $dsn, $user, $password) = @_;

    my %BLST_reslt = ( );

    my $dbh = DBI->connect($dsn, $user, $password,
                          { RaiseError => 1, AutoCommit => 0 });

    my $pfamQ = $dbh->prepare("select pfam_num from pfam2cog where cog_num = ?");
    my $GOq = $dbh->prepare("SELECT go.go_num, go.go_desc FROM pfam
                            JOIN go2pfam ON pfam.pfam_num = go2pfam.pfam_num
                            JOIN go ON go2pfam.go_num = go.go_num
                            WHERE pfam.pfam_num = ?");

    my $searchin = new Bio::SearchIO(-format => 'blastxml',
                                     -file => $xml_file);

    my $result = $searchin->next_result();

    while( my $hit = $result->next_hit() )
    {
        #print "Hit: ", $hit->description(), "\n";
        my $hit_desc = $hit->description();
        my @hit_title = split(/\s/, $hit->description(),3);
        #print "CDD: ", $hit_title[0], "\tLABEL: ", $hit_title[1], "\tDESC: ", $hit_title[2], "\n";
        my $db_def = substr $hit_title[0], 0, 3;
        #print "Codex: ", uc $db_def, "\n";
        switch (uc $db_def)
        {
            case ('COG')
            {
                if (not exists $BLST_reslt{COG}{CD})
                {
                    $BLST_reslt{COG}{CD} = $hit_title[0];
                }
            }
        }
    }
}

```

```

        $BLST_reslt{COG}{LABEL} = $hit_title[1];
        my @desc_func = split(/\s\{(.*)\}/,$hit_title[2],2);
        $BLST_reslt{COG}{DESC} = $desc_func[0];
        $BLST_reslt{COG}{FUNC} = $desc_func[1];
        my $hsp = $hit->next_hsp();
        $BLST_reslt{COG}{EVALUE} = $hsp->evaluate();
        my $src = $pfamQ->execute($hit_title[0]) or die "Can't execute statement: $DBI::errstr";
        my $pfam_des = $pfamQ->fetchrow_array;
        $BLST_reslt{COG}{PFAM} = $pfam_des;
        my $rft = $GOq->execute($pfam_des) or die "Can't execute statement: $DBI::errstr";
        my ($go_num, $go_des) = $GOq->fetchrow_array;
        $BLST_reslt{COG}{GO} = $go_num;
        $BLST_reslt{COG}{GO_DESC} = $go_des;
    }
}
case ('PFA')
{
    if (not exists $BLST_reslt{PFAM}{CD})
    {
        $BLST_reslt{PFAM}{CD} = join('','PF', substr ($hit_title[0],4));
        $BLST_reslt{PFAM}{LABEL} = $hit_title[1];
        my @func = split(/\s\{(.*)\}/,$hit_title[2],2);
        $BLST_reslt{PFAM}{DESC} = $func[0];
        my $hsp = $hit->next_hsp();
        $BLST_reslt{PFAM}{EVALUE} = $hsp->evaluate();
        my $rft = $GOq->execute($BLST_reslt{PFAM}{CD}) or die "Can't execute statement: $DBI::errstr";
        my ($go_num, $go_des) = $GOq->fetchrow_array;
        $BLST_reslt{PFAM}{GO} = $go_num;
        $BLST_reslt{PFAM}{GO_DESC} = $go_des;
    }
}
}
}
$pfamQ->finish;
$GOq->finish;
$dbh->disconnect;
return \%BLST_reslt;
}

sub CDD_printResults
{
    my ($shashA) = @_ ;
    my %aaLine = %$shashA;
    for my $family ( keys %aaLine ) {
        print "Family:\n";
        for my $role ( keys %{ $aaLine{$family} } ) {
            print "$role=$aaLine{$family}{$role}";
        }
        print "\n";
    }
}

sub HTML_aa3to1 {
    my($input) = @_ ;

    my %three2one = (
        'ALA' => 'A',
        'VAL' => 'V',
        'LEU' => 'L',
        'ILE' => 'I',
        'PRO' => 'P',
        'TRP' => 'W',
        'PHE' => 'F',
        'MET' => 'M',
        'GLY' => 'G',
        'SER' => 'S',
        'THR' => 'T',
        'TYR' => 'Y',
        'CYS' => 'C',
        'ASN' => 'N',
        'GLN' => 'Q',
        'LYS' => 'K',
        'ARG' => 'R',
        'HIS' => 'H',
        'ASP' => 'D',
        'GLU' => 'E',
    );

    # clean up the input
    $input =~ s/\n/ /g;

    my $seq = '';

    # This use of split separates on any contiguous whitespace
    my @code3 = split(' ', $input);

    foreach my $code (@code3) {
        # A little error checking
        if(not defined $three2one{$code}) {

```

```

        print "Code $code not defined\n";
        next;
    }
    $seq .= $three2one{$code};
}
return $seq;
}

sub HTML_CSAREqFile
{
    my ($pdb_id) = @_;
    #my ($pdb_str,$chn_id) = split(/_/, $pdb_id);

    @CDD::ISA = qw(HTML::Parser);

    my $URL = 'http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/CSA/CSA_Site_Wrapper.pl?pdb='.$pdb_id;
    my $content = get($URL);
    my $parser = CDD->new;
    $parser->parse($content);
    my $txt_out = $parser->{TEXT};
    for ($txt_out)
    {
        s/^\s+//;
        s/^\s+$//;
    }
    my $filename = "./pdb/".$pdb_id.".pdb";

    open(OUTFILE, ">".$filename) or die "Can't open filename $filename";
    print OUTFILE $txt_out;
    close OUTFILE;

    return $filename;
}

sub CSA_loadDataStruct
{
    my ($fileName,$lowScoreCutoff) = @_;
    $fileName =~ s/^\s+//;
    my $file_contents = read_file($fileName);
    my @multMatches = split(/>{4}/, $file_contents);
    my %fileRec;
    foreach my $myProt (@multMatches)
    {
        my @subjNot = ($myProt =~ m/([a-zA-z0-9]+)/);

        my @singleMatches = split(/\n/, $myProt);

        foreach my $oneMatch (@singleMatches)
        {
            if (!$oneMatch =~ /^$/)
            {
                #print $oneMatch."\n";
                my @oneMatchData = split(/\t/, $oneMatch);
                #print "\t\t".$oneMatchData[4]."\n";
                if (((($oneMatchData[6]-$oneMatchData[5])/($oneMatchData[5]))>$lowScoreCutoff)
                {
                    $fileRec{$oneMatchData[0]}{$oneMatchData[4]}{$oneMatchData[3]}{"Tc"}=$oneMatchData[5];
                    $fileRec{$oneMatchData[0]}{$oneMatchData[4]}{$oneMatchData[3]}{"T"}=$oneMatchData[6];
                    $fileRec{$oneMatchData[0]}{$oneMatchData[4]}{$oneMatchData[3]}{"NormScore"}=((($oneMatchData[6]-
                    $oneMatchData[5])/($oneMatchData[5]));
                }
            }
        }
    }
    return \%fileRec;
}

sub CSA_readParams
{
    my ($paramFile) = @_;
    my $fileConts = read_file($paramFile);
    my %config = $fileConts =~ /(\w+)>\s(.+)/mg ;
    foreach my $myDir (keys %config)
    {
        $config{$myDir} =~ s/^\s+//;
    }
    return \%config;
}

sub CSA_readTrueVals
{
    my ($trueValFile) = @_;
    my $fileConts = read_file($trueValFile);
    my %trueVals;
    my @truePairs = $fileConts =~ /^[>F]+\s+(\S+)\s+(\S+).+/mg ;
    while (scalar @truePairs > 0)
    {
        $trueVals{shift(@truePairs)}{shift(@truePairs)}= ();
    }
}

```

```

    }
    delete ($trueVals{' '});
    delete ($trueVals{'0'});
    return \%trueVals;
}

sub CSA_UniProt_DB
{
    my ($uniprotDir, $dbName) = @_;
    $uniprotDir =~ s/\s+//;
    $dbName =~ s/\s+//;

    my $db = Bio::DB::Flat->new(-directory => $uniprotDir,
                                -dbname => $dbName,
                                -write_flag => 0,
                                -index => 'bdb',
                                -format => 'fasta');

    return \%db;
}

sub CSA_runActiveSiteAlign
{
    my ($pfile,$sfile,$siteFile,$myResults) = @_;
    # $pfile =~ s/\s+//;
    # $sfile =~ s/\s+//;
    # $siteFile =~ s/\s+//;

    #open(outScore, ">$pfile.score") or die "Can't open Output File: $!";

    if ($^O eq "MSWin32"){
        system "clustalw /profile1=\"$pfile\" /profile2=\"$sfile\" /sequences /outfile=aln$$$.aln >./null$$ 2>&1";
    }
    else {
        system "./clustalw -profile1=$pfile -profile2=$sfile -sequences -outfile=aln$$$.aln >./null$$ 2>&1";
    }
    # system "rm pid$$$.aln";
}

my ($seqref, $refid)=readClustalW($pfile);
my ($seq,$seqid)=readClustalW("aln$$$.aln");

my @refkeys=keys %$seqref;
my @seqkeys=keys %$seq;
my $len=length($seq->{$seqid->{0}});
my $nodel;
my @dels=();
#print "num refkeys= $#refkeys, len= $len\n";
#print "@refkeys\n";
#print "@seqkeys\n";

for(my $i=0; $i<$len; $i++) {
    $nodel=0;
    foreach my $k (@refkeys) {
        if(not exists $seq->{$k}) {
            print "The enzyme $seqref->{$k} is not in the heterogenous set!\n";
        }
        if(substr($seq->{$k},$i,1) ne "-") {
            $nodel=1;
            last;
        }
    }
    push @dels,$i if($nodel==0);
}
#Remove complete insertion position from the end to back to the begin of a string
my $numdels=scalar @dels;
my $numseq=scalar @seqkeys;
for(my $i=0; $i<$numdels; $i++) {
    my $idx=pop @dels;
    foreach my $k(@seqkeys) {
        substr($seq->{$k},$idx,1)=''; #delete a position at $idx
    }
}
my $st=0;
my $thekey;
my $pos="1234567890";
my $npos=0;
my %sites=();

open(FIDP, '<',$siteFile) or die "Can not open the active site file\n";
while(<FIDP>) { #read in active sites indexed by the position number
    if(m/POS>\s*(\d+)\s*(\w)/) {
        if(not exists $sites{$1}) {
            $sites{$1}=[uc($2)];
        } else {
            push @{$sites{$1}}, uc($2);
        }
        #insertAllowedMutation($sites{$1});
    }
}

```

```

    }
}
close(FIDP);
foreach my $ky (keys %sites) {
    insertAllowedMutation($sites{$ky});
}
#Search active sites for each sequence provided by $sfile
my $poscount = 0;
my @sitelst = ();
if (lc (substr $sfile, (length $sfile)-3) eq 'csa' )
{
    my $seqIn = new Bio::AlignIO(-format => 'clustalw',
        -file => "aln$$aln");
    my $aln = $seqIn->next_aln;
    $$myResults .= $aln->no_sequences();

    my $refIn = new Bio::AlignIO(-format => 'clustalw',
        -file => $pfile);
    my $refAln = $refIn->next_aln;
    $$myResults .= $refAln->no_sequences();

    @sitelst=sort {$a <=> $b} keys(%sites);
    foreach my $seq ( $aln->each_seq() ) {
        my $seqname=$seq->display_name();
        $poscount=0;
        # Problem if sequence already in the alignment!!!!
        next if(exists $seqref->{$seqname});
        $$myResults .= "NAME> $seqname\n";
        foreach my $st(@sitelst) {
            my $site_loc = $aln->column_from_residue_number( $seqname, $st );
            my $aa = $seq->subseq($site_loc,$site_loc);
            my $out=siteCheck($sites{$st}, $aa);
            $$myResults .= "POS> $st $aa $out ".join(" ",@{$sites{$st}})."\n";
            $poscount++ if($out ne "-");
        }
        $$myResults .= "STA> $poscount ".scalar(@sitelst)."\n\n";
    }
}
else
{
    my $numseq=scalar(@seqkeys);
    @sitelst=sort {$a <=> $b} keys(%sites);
    for(my $i=0; $i<$numseq; $i++) {
        my $seqname=$seqid->{$i};
        $poscount=0;
        next if(exists $seqref->{$seqname});
        $$myResults .= "NAME> $seqname\n";
        foreach my $st(@sitelst) {
            my $aa=substr($seq->{$seqname},$st-1,1);
            my $out=siteCheck($sites{$st}, $aa);
            $$myResults .= "POS> $st $aa $out ".join(" ",@{$sites{$st}})."\n";
            $poscount++ if($out ne "-");
        }
        $$myResults .= "STA> $poscount ".scalar(@sitelst)."\n\n";
    }
}
#close outScore;
if ($^O eq "MSWin32"){
    `del aln$$aln`;
}
else{
    `rm aln$$aln`;
}
return ($poscount,scalar(@sitelst));
}

sub readClustalW() {
    my $fname=$_[0];
    my $fin;
    my @lnlist;
    my %out;
    my %nameid;
    my $seqName;
    my $ct=0; #the position index of the seugence in the file.
    open($fin,"<$fname") or die "Can not open the ClustalW output file $fname.\n";
    while(<$fin>) {
        next if(m/^\s*/); #skip the empty line
        next if(m/clustal/i); #skip the header
        next if(m/^\s{10,}/); #skip the line with at least 10 leading blanks
        @lnlist=split(/\s+/, $_);
        $lnlist[0]=~m/(^w+)/; # ~m/(^w+_w+)/;
        $seqName=$1;
        if(exists $out{$seqName}) {
            chomp($lnlist[1]);
            $out{$seqName}=$out{$seqName}.$lnlist[1];
        }
        else {
            $out{$seqName}=$lnlist[1];
            $nameid{$ct}=$seqName;
            $ct++;
        }
    }
}

```

```

    }
    return(\%out, \%nameid);
}

sub insertAllowedMutation {
    my $site=shift;
    foreach my $st(@{$site}) {
        if($st=~m/D|R/) { push @{$site}, ('d','r'); }
        elsif($st=~m/K/) {push @{$site}, 'h'; }
    }
}

sub siteCheck {
    my $site=shift;
    my $aa=shift;
    $aa=uc($aa);
    my $aal=lc($aa);
    my $out="-"; #the active site is not found
    foreach my $st(@{$site}) {
        return("+") if($aa eq $st);           #the active site exists in the searched sequence
        $out="?" if($aal eq $st);           #the allowed mutation at the active site exists
    }
    return $out;
}

1;

```

## Appendix D

```

#!/usr/bin/perl -w
use strict;
use warnings;

use Bio::Seq;
use Bio::SeqIO;
use Bio::DB::SwissProt;
use File::Slurp;
use Getopt::Std;
use POSIX qw(strftime);
#use String::Approx 'amatch';

use CDD;

my %opts=();
getopts('s:l',\%opts);
# Process file that contains various paths to the directories where profiles, active sites, etc. are stored
# Returns a hash with CODE:Directory structure
my $paramPtr = CDD::CSA_readParams($ARGV[0]);

#print $gbSeq->seq;

my $trueValPtr = CDD::CSA_readTrueVals($paramPtr->{TRUE});
my $trueVal = %$trueValPtr;

# Returns a data structure (hash of hashes of hashes) of the matched profiles
# arg1: file of matched profiles; arg2: Lower normalized threshold of match score
my $fileRecPtr = CDD::CSA_loadDataStruct($paramPtr->{OUT},0.0);
my %fileRec = %$fileRecPtr;

# Load "skipped" proteins into hash to double check them
#
my %is_skipped = ();
if(exists $opts{s}) {
    eval {
        my @rejectedNames = read_file($opts{s});
        for (@rejectedNames) {$is_skipped{$_} = 1}
    };
}

my $gb = ();
if(exists $opts{l}) {
    # Setup SwissProt database query service to the local database
    # First param is the directory where DB is located, second param is the name of the DB
    my $db_ptr = CDD::CSA_UniProt_DB($paramPtr->{UNI},$paramPtr->{DB});
    $gb = $$db_ptr;
} else {
    # Setup SwissProt database query service to the US location
    $gb = new Bio::DB::SwissProt(-servertype => 'expasy',
                                -hostlocation => 'us');
}
#$gb->request_format("fasta");

#my $out = Bio::SeqIO->new( -file => '>SEVafasta.fasta', -format => 'fasta');
#$out->preferred_id_type('display');
#my $mySeqOb = $gb->get_Seq_by_id('lA11_ORYSA'); # Unique ID
#$out->write_seq($mySeqOb);

my $be4break = 1;
my $mySeqOb = ();
my $reportFilename = "Report".strftime("%Y%m%d_%H%M", localtime).'.rpt';
write_file( "./Report/".$reportFilename,
"QUERY_NAME\tEC_NUMBR\tSITE_HITS\tTOTAL_SITES\tPCNT_MATCH\tEC_HITS\tTOTAL_NORM\tAVG_NORM\tMAX_NORM\tV\tTRUE_MATCH\n\n"
);

# Process each unknown sequence
foreach my $unknown (sort keys %fileRec)

```

```

{
  $@ = ();
  # Device to start processing from certain point if previous execution crashed
  # Mostly used when the SwissProt notation has been changed or retired which
  # causes execution to crash during writing of retrieved sequence to a Fasta file
  # =====
  #if ($unknown eq 'MTRF_METBF')
  #{
  #   #unknown = 'RPOBC_HELHP';
  #   $be4break = 0;
  #}
  #next if ($be4break == 1);
  # =====
  # The following statement checks the $unknown against a list of values
  # that have been previously not processed to doublecheck that their names have changed
  #
  if(exists $opts{s}) {
    next unless exists $is_skipped{$unknown."\"n\"};
  }

  print $unknown."\"n\"";

  # Initialize the results string
  my $myResults = ();
  my %unknownTotals = ();
  $unknownTotals{'printReport'} = sub {
    my $unknownTotString;
    my $hitPercentage;
    foreach my $k (keys %{$unknownTotals{'EC'}}) {
      @{$unknownTotals{'EC'}{$k}{AllNorms}} = sort {$b <=> $a} @{$unknownTotals{'EC'}{$k}{AllNorms}};
      eval {$hitPercentage =
sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalHits}/$unknownTotals{'EC'}{$k}{TotalSites});};
      if ($@) {$hitPercentage = " N\\A\";";
        $unknownTotString .= $unknown."\"t\"
          .$k."\"t\"
          .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalHits})."\"t\"
          .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalSites})."\"t\"
          . $hitPercentage."\"t\"
          .sprintf("%04d", $unknownTotals{'EC'}{$k}{Count})."\"t\"
          .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm})."\"t\"

        .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm}/$unknownTotals{'EC'}{$k}{Count})."\"t\"
          .sprintf("%.4f", $unknownTotals{'EC'}{$k}{AllNorms}[0])."\"t\"
          .((join ("\"t\", keys %{$trueVal{$unknown}})) =~ /$k/) ? "1" : "0")."\"t\"
          .join ("\"t\", keys %{$trueVal{$unknown}})."\"n\";
      }
      $unknownTotString .= ">>>>\"n\";
      return $unknownTotString;
    };
    $unknownTotals{'printECTotals'} = sub {
      my $ECTotString;
      my $hitPercentage;
      $ECTotString .= "EC_NUMBR\tSITE_HITS\tTOTAL_SITES\tPCNT_MATCH\tEC_HITS\tTOTAL_NORM\tAVG_NORM\tMAX_NORM\tV\"n\";
      foreach my $k (keys %{$unknownTotals{'EC'}}) {
        @{$unknownTotals{'EC'}{$k}{AllNorms}} = sort {$b <=> $a} @{$unknownTotals{'EC'}{$k}{AllNorms}};
        eval {$hitPercentage =
sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalHits}/$unknownTotals{'EC'}{$k}{TotalSites});};
        if ($@) {$hitPercentage = " N\\A\";";
          $ECTotString .= $k."\"t\"
            .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalHits})."\"t\"
            .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalSites})."\"t\"
            . $hitPercentage."\"t\"
            .sprintf("%04d", $unknownTotals{'EC'}{$k}{Count})."\"t\"
            .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm})."\"t\"

          .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm}/$unknownTotals{'EC'}{$k}{Count})."\"t\"
            .sprintf("%.4f", $unknownTotals{'EC'}{$k}{AllNorms}[0])."\"t\"
            .((join ("\"t\", keys %{$trueVal{$unknown}})) =~ /$k/) ? "1" : "0")."\"n\";
        }
      }
      return $ECTotString;
    };
    # Obtain sequence for the unknown protein from local DB
    #my $mySeqOb = $$db_ptr->get_seq_by_id($unknown);

    # Obtain sequence for the unknown protein from Internet
    # Skip the unknown protein if its designation no longer exists in SwissProt
    # and add it to @changed array
    eval {$mySeqOb = $gb->get_seq_by_id($unknown);}; # Unique ID
    if ($@)
    {
      write_file( 'changed.prots', { append => 1 }, $unknown."\"n\" );
      next;
    }
    # Setup filename for the output of unknown sequence in Fasta format
    my $out = Bio::SeqIO->new( -file => '> '.$paramPtr->{ROOT}. $unknown."\".fasta", -format => 'fasta');
    # Output Display name ('XXX_XXXX') comes first
    $out->preferred_id_type('display');
    # Write sequence to file
    $out->write_seq($mySeqOb);
  }
}

```

```

# For each profile that aligned to unknown sequence at greater than specified normalized score
foreach my $known (sort keys %{$fileRec{$sunknown}})
{
  # Separate profiles by EC number in case same known sequence profile belongs to 2 different EC numbers
  foreach my $ecnum (sort keys %{$fileRec{$sunknown}{$known}})
  {
    # Check to see if accuracy of the active site profile is greater than 80%
    $myResults .= "MATCH> $known\n";
    $myResults .= "EC> $ecnum\n";
    $myResults .= "NORM> $fileRec{$sunknown}{$known}{$ecnum}{NormScore}\n";
    my $fisPtr = CDD::CSA_readParams("$paramPtr->{FIS}$ecnum/$ecnum.$known.fis");
    if ($fisPtr->{ACC} < 0.80)
    {
      $myResults .= "ACC> $fisPtr->{ACC} < 0.80!!!!\n";
      $unknownTotals{'EC'}{$ecnum}{TotalSites} += 0;
      $unknownTotals{'EC'}{$ecnum}{TotalHits} += 0;
      $myResults .= "STA> N\\A N\\A\n";
    }
    else
    {
      # Align unknown sequence to the matched profile, check active sites, and return the results
      ($fileRec{$sunknown}{$known}{$ecnum}{SiteHits}, $fileRec{$sunknown}{$known}{$ecnum}{TotalSites}) =
      CDD::CSA_runActiveSiteAlign("$paramPtr->{ALN}$ecnum/$ecnum.$known.aln", $paramPtr->{ROOT}.$sunknown.".fasta",
      "$paramPtr->{FIS}$ecnum/$ecnum.$known.fis", $myResults);
      $unknownTotals{'EC'}{$ecnum}{TotalSites} += $fileRec{$sunknown}{$known}{$ecnum}{TotalSites};
      $unknownTotals{'EC'}{$ecnum}{TotalHits} += $fileRec{$sunknown}{$known}{$ecnum}{SiteHits};
    }
    $unknownTotals{'EC'}{$ecnum}{Count} += 1;
    $unknownTotals{'EC'}{$ecnum}{TotalNorm} += $fileRec{$sunknown}{$known}{$ecnum}{NormScore};
    push @{$unknownTotals{'EC'}{$ecnum}{AllNorms}}, $fileRec{$sunknown}{$known}{$ecnum}{NormScore};
  }
}
# Output results for each unknown sequence to separate file
write_file( "./Score3/" . $sunknown . ".score", $myResults );
append_file( "./Score3/" . $sunknown . ".score", $unknownTotals{'printECTotals'}->() );
append_file( "./Report/" . $reportFilename, $unknownTotals{'printReport'}->() );
append_file( "./Score3/" . $sunknown . ".score", "QUERY> $sunknown\tTRUE> ".join ("\t", keys %{$trueVal{$sunknown}}) . "\n"
);
# Reset the results string;
$myResults = ();
# Erase Fasta file of unknown sequence
# ***Currently Does Not Erase!***
unlink($paramPtr->{ROOT}.$sunknown.".fasta"); #or die "could not unlink: $!";
unlink($paramPtr->{ROOT}.$sunknown.".dnd"); #or die "could not unlink: $!";
}

exit;

```

## Appendix E

```
#!/usr/bin/perl -w
use strict;
use warnings;

use Bio::Seq;
use Bio::SeqIO;
use Bio::DB::SwissProt;
use File::Slurp;
use Getopt::Std;
use POSIX qw(strftime);

use CDD;

my %opts=();
getopts('s:l',\%opts);
# Process file that contains various paths to the directories where profiles, active sites, etc. are stored
# Returns a hash with CODE:Directory structure
my $paramPtr = CDD::CSA_readParams($ARGV[0]);

# Load True values from the .sta file
my $trueValPtr = CDD::CSA_readTrueVals($paramPtr->{TRUE});
my $trueVal = %$trueValPtr;

# Returns a data structure (hash of hashes of hashes) of the matched profiles
# arg1: file of matched profiles; arg2: Lower normalized threshold of match score
my $fileRecPtr = CDD::CSA_loadDataStruct($paramPtr->{OUT},0.0);
my $fileRec = %$fileRecPtr;

# Load "skipped" proteins into hash to double check them
#
my %is_skipped = ();
if(exists $opts{s}) {
    eval {
        my @rejectedNames = read_file($opts{s});
        for (@rejectedNames) {$is_skipped{$_} = 1}
    };
}

my $gb = ();
if(exists $opts{l}) {
    # Setup SwissProt database query service to the local database
    # First param is the directory where DB is located, second param is the name of the DB
    my $db_ptr = CDD::CSA_UniProt_DB($paramPtr->{UNI},$paramPtr->{DB});
    $gb = $$db_ptr;
} else {
    # Setup SwissProt database query service to the US location
    $gb = new Bio::DB::SwissProt(-servertime => 'expasy',
                                -hostlocation => 'us');
}

my $be4break = 1;
my $mySeqOb = ();
my $reportFilename = "Report".strftime("%Y%m%d_%H%M", localtime).'.rpt';
write_file( "./Report4d/".$reportFilename,
"QUERY_NAME\tEC_NMBR\tHITS\tSITES\tPERCENT\tEC_#\tT_NORM\tA_NORM\tM_NORM\tV\tTRUE_MATCH\n\n" );

foreach my $unknown (sort keys %fileRec)
{
    $@ = ();
    # Device to start processing from certain point if previous execution crashed
    # Mostly used when the SwissProt notation has been changed or retired which
    # causes execution to crash during writing of retrieved sequence to a Fasta file
    # =====
    if ($unknown eq 'MAK_HUMAN')
    {
        {
            # $unknown = 'RPOBC_HELHP';
            $be4break = 0;
        }
    }
    next if ($be4break == 1);
    # =====
}
```

```

# The following statement checks the $unknown against a list of values
# that have been previously not processed to doublecheck that their names have changed
#
if(exists $opts{s}) {
  next unless exists $is_skipped{$unknown."\n"};
}

print $unknown."\n";

# Initialize the results string
my $myResults = ();
my %unknownTotals = ();
$unknownTotals{'printReport'} = sub {
  my $unknownTotString;
  my $hitPercentage;
  foreach my $k (keys %{$unknownTotals{'EC'}}) {
    @{$unknownTotals{'EC'}{$k}{AllNorms}} = sort {$b <=> $a} @{$unknownTotals{'EC'}{$k}{AllNorms}};
    eval {$hitPercentage =
sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalHits}/$unknownTotals{'EC'}{$k}{TotalSites});};
    if ($@) {$hitPercentage = " N\\A";}
    $unknownTotString .= $unknown."\t"
      .$.."\t"
      .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalHits})."\t"
      .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalSites})."\t"
      .$hitPercentage."\t"
      .sprintf("%04d", $unknownTotals{'EC'}{$k}{Count})."\t"
      .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm})."\t"
      .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm}/$unknownTotals{'EC'}{$k}{Count})."\t"
      .sprintf("%.4f", $unknownTotals{'EC'}{$k}{AllNorms}[0])."\t"
      .((join ("\t", keys %{$trueVal{$unknown}})) =~ /$k/ ? "1" : "0")."\t"
      .join ("\t", keys %{$trueVal{$unknown}})."\n";
    }
    $unknownTotString .= ">>>>\n";
  }
  return $unknownTotString;
};
$unknownTotals{'printECTotals'} = sub {
  my $ECTotString;
  my $hitPercentage;
  $ECTotString .= "EC_NMBR\tHITS\tSITES\tPERCENT\tEC_#\tT_NORM\tA_NORM\tM_NORM\tV\n";
  foreach my $k (keys %{$unknownTotals{'EC'}}) {
    @{$unknownTotals{'EC'}{$k}{AllNorms}} = sort {$b <=> $a} @{$unknownTotals{'EC'}{$k}{AllNorms}};
    eval {$hitPercentage =
sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalHits}/$unknownTotals{'EC'}{$k}{TotalSites});};
    if ($@) {$hitPercentage = " N\\A";}
    $ECTotString .= $.."\t"
      .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalHits})."\t"
      .sprintf("%04d", $unknownTotals{'EC'}{$k}{TotalSites})."\t"
      .$hitPercentage."\t"
      .sprintf("%04d", $unknownTotals{'EC'}{$k}{Count})."\t"
      .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm})."\t"
      .sprintf("%.4f", $unknownTotals{'EC'}{$k}{TotalNorm}/$unknownTotals{'EC'}{$k}{Count})."\t"
      .sprintf("%.4f", $unknownTotals{'EC'}{$k}{AllNorms}[0])."\t"
      .((join ("\t", keys %{$trueVal{$unknown}})) =~ /$k/ ? "1" : "0")."\n";
    }
  }
  return $ECTotString;
};
# Obtain sequence for the unknown protein from local DB
#my $mySeqOb = $$db_ptr->get_seq_by_id($unknown);

# Obtain sequence for the unknown protein from Internet
# Skip the unknown protein if its designation no longer exists in SwissProt
# and add it to @changed array
eval {$mySeqOb = $gb->get_seq_by_id($unknown);}; # Unique ID
if ($@)
{
  write_file( 'changed.prots', { append => 1 }, $unknown."\n" );
  next;
}
# Setup filename for the output of unknown sequence in Fasta format
my $out = Bio::SeqIO->new( -file => '> '.$paramPtr->{ROOT}. $unknown.".fasta", -format => 'fasta');
# Output Display name ('XXX_XXXXX') comes first
$out->preferred_id_type('display');
# Write sequence to file
$out->write_seq($mySeqOb);

# For each profile that aligned to unknown sequence at greater than specified normalized score
foreach my $known (sort keys %{$fileRec{$unknown}})
{
  # Separate profiles by EC number in case same known sequence profile belongs to 2 different EC numbers
  foreach my $ecnum (sort keys %{$fileRec{$unknown}{$known}})
  {
    my ($myUnknown, $thisAlignment) = CDD::get4dAlignment("$paramPtr->{ALN}$ecnum/$ecnum.$known.aln0",
$paramPtr->{ROOT}. $unknown.".fasta");
    my $psblFisMatches = CDD::getMatchFIS($paramPtr->{FIS}, $ecnum, $known);

    foreach my $fisMatch (keys %$psblFisMatches)
    {

```



# Appendix F

```

MATCH> 3BHS1_MESAU
EC> 1.1.1
NDRHD> 7.8742961952657
NAME> 3BHS4_RAT
PDS> 74 F + F
PDS> 389 T + T
STA> 2 2

MATCH> 3BHS1_MOUSE
EC> 5.3.3
NDRHD> 10.6712328767123
NAME> 3BHS4_RAT
PDS> 186 T + T
PDS> 164 E + E
PDS> 268 D + D d r
PDS> 342 V + V
PDS> 358 W + W
STA> 5 5

MATCH> 3BHS_UACCA
EC> 5.3.3
NDRHD> 9.95394736842105
NAME> 3BHS4_RAT
PDS> 165 E + E
PDS> 307 S + S
PDS> 308 K + K h
STA> 3 3

MATCH> ALD2_SPOSA
EC> 1.1.1
NDRHD> 3.02539682539683
NAME> 3BHS4_RAT
PDS> 44 T + T
PDS> 49 F + F
PDS> 97 D + D d r
PDS> 118 H + H
PDS> 151 C + C
PDS> 295 H + H
STA> 6 6

MATCH> DFRA_ARATH
EC> 1.1.1
NDRHD> 3.02539682539683
NAME> 3BHS4_RAT
PDS> 44 T + T
PDS> 49 F + F
PDS> 97 D + D d r
PDS> 118 H + H
PDS> 151 C + C
PDS> 295 H + H
STA> 6 6

MATCH> DFRA_PETHV
EC> 1.1.1
NDRHD> 3.07051282051282
NAME> 3BHS4_RAT
PDS> 44 T + T
PDS> 49 F + F
PDS> 97 D + D d r
PDS> 118 H + H
PDS> 151 C + C
PDS> 295 H + H
STA> 6 6

MATCH> FCL_CRIGR
EC> 1.1.1
NDRHD> 2.1038961038961
ACC> 0.295455 < 0.80!!!!
STA> N/A N/A

MATCH> NUEN_BOVIN
EC> 1.6.5
NDRHD> 0.1527777777777778
NAME> 3BHS4_RAT
PDS> 57 U - T
PDS> 123 A - S
PDS> 321 S - R d r
STA> 0 3

MATCH> RFBD_RHISH
EC> 1.1.1
NDRHD> 0.901315789473684
NAME> 3BHS4_RAT
PDS> 389 T + T
STA> 1 1

MATCH> RFBD_SHIFL
EC> 1.1.1
NDRHD> 0.117088607594937
NAME> 3BHS4_RAT
PDS> 96 D + D d r
PDS> 139 N + N
PDS> 168 G - F
PDS> 192 V + V
PDS> 196 K + K h
PDS> 280 E + E
STA> 5 6

```

EC_NUMBR	SITE_HITS	TOTAL_SITES	PCNT_MATCH	EC_HITS	TOTAL_NORM	AUG_NDRH	MAX_NDRH	U
5.3.3	0008	0008	1.0000	0002	20.6252	10.3126	10.6712	1
1.6.5	0000	0000	0.0000	0001	0.1528	0.1528	0.1528	0
1.1.1	0026	0027	0.9630	0007	20.1180	2.8740	7.8744	1
QUERY>	3BHS4_RAT	TRUE	1.1.1.	5.3.3.				

## Appendix G

### catsitelist

#### Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
site_num	int(10) unsigned	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	Site Number
pdb_num	char(4)	utf8_general_ci	NO				select,insert,update,references	PDB number
chain_id	char(1)	utf8_general_ci	YES		(NULL)		select,insert,update,references	PDB chain ID
postn_num	int(4) unsigned	(NULL)	NO				select,insert,update,references	site Uniprot position
aa_id	char(1)	utf8_general_ci	NO				select,insert,update,references	Amino Acid 1-letter code

#### Indexes

Table	Non unique	Key name	Seq in index	column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
catsitelist	0	PRIMARY	1	site_num	A	4534	(NULL)	(NULL)		BTREE	

### cog

#### Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
cog_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
cog_desc	varchar(100)	latin1_swedish_ci	YES	MUL	(NULL)		select,insert,update,references	

#### Indexes

Table	Non unique	Key name	Seq in index	column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
cog	0	PRIMARY	1	cog_num	A	4685	(NULL)	(NULL)		BTREE	
cog	1	Index_desc	1	cog_desc	A	4685	(NULL)	(NULL)	YES	BTREE	

### cog2prosite

#### Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
cog_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
prosite_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	

#### Indexes

Table	Non unique	Key name	Seq in index	column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
cog2prosite	0	PRIMARY	1	cog_num	A	3858	(NULL)	(NULL)		BTREE	
cog2prosite	0	PRIMARY	2	prosite_num	A	3858	(NULL)	(NULL)		BTREE	

#### Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
FK_gomap_cog_cog2prosite	cog	`cog_num`	`cog_num`	ON DELETE CASCADE ON UPDATE CASCADE

**csa\_ref****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
rec_num	int(10) unsigned	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	record number
ec_num	varchar(20)	utf8_general_ci	YES		(NULL)		select,insert,update,references	EC number
uniprot_not	varchar(20)	utf8_general_ci	YES		(NULL)		select,insert,update,references	UNIPROT designation
pdb_num	char(4)	utf8_general_ci	NO				select,insert,update,references	PDB notation

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
csa_ref	0	PRIMARY	1	rec_num	A	964	(NULL)	(NULL)		BTREE	

**go****Fields**

Field	Type	collation	Null	Key	Default	Extra	Privileges	Comment
go_num	varchar(7)	utf8_general_ci	NO	PRI			select,insert,update,references	
go_desc	varchar(255)	utf8_general_ci	YES	MUL	(NULL)		select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
go	0	PRIMARY	1	go_num	A	4890	(NULL)	(NULL)		BTREE	
go	1	Index_go_desc	1	go_desc	A	4890	(NULL)	(NULL)	YES	BTREE	

**go2cog****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
go_num	varchar(7)	utf8_general_ci	NO	MUL			select,insert,update,references	
cog_letter	varchar(50)	utf8_general_ci	YES	MUL	(NULL)		select,insert,update,references	
cog_desc	varchar(50)	utf8_general_ci	YES	MUL	(NULL)		select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
go2cog	1	Index_COG_desc	1	cog_desc	A	99	(NULL)	(NULL)	YES	BTREE	
go2cog	1	Index_go	1	go_num	A	99	(NULL)	(NULL)		BTREE	
go2cog	1	Index_COG_letter	1	cog_letter	A	49	(NULL)	(NULL)	YES	BTREE	

**Foreign Key Relationships**

FK Id	Reference Table	Source Column	Target Column	Extra Info
FK_gomap_Go_go2cog	go	`go_num`	`go_num`	ON DELETE CASCADE ON UPDATE CASCADE

**go2ec****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
go_num	varchar(7)	utf8_general_ci	NO	MUL			select,insert,update,references	
ec_num	varchar(50)	utf8_general_ci	NO	MUL			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
go2ec	1	Index_go	1	go_num	A	3206	(NULL)	(NULL)		BTREE	
go2ec	1	Index_ec	1	ec_num	A	3206	(NULL)	(NULL)		BTREE	

**Foreign Key Relationships**

FK Id	Reference Table	Source Column	Target column	Extra Info
FK_gomap_Go_go2ec	go	`go_num`	`go_num`	ON DELETE CASCADE ON UPDATE CASCADE

**go2pfam****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
go_num	varchar(50)	utf8_general_ci	NO	PRI			select,insert,update,references	
pfam_num	varchar(7)	utf8_general_ci	NO	PRI			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
go2pfam	0	PRIMARY	1	go_num	A	4134	(NULL)	(NULL)		BTREE	
go2pfam	0	PRIMARY	2	pfam_num	A	8268	(NULL)	(NULL)		BTREE	
go2pfam	1	Index_go	1	go_num	A	4134	(NULL)	(NULL)		BTREE	
go2pfam	1	Index_pfam	1	pfam_num	A	8268	(NULL)	(NULL)		BTREE	

**Foreign Key Relationships**

FK Id	Reference Table	Source Column	Target column	Extra Info
FK_gomap_Go_go2pfam	go	`go_num`	`go_num`	ON DELETE CASCADE ON UPDATE CASCADE,
FK_gomap_Pfam_go2pfam	pfam	`pfam_num`	`pfam_num`	ON DELETE CASCADE ON UPDATE CASCADE

**go2prodom****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
go_num	varchar(50)	utf8_general_ci	NO	MUL			select,insert,update,references	
prodom_num	varchar(50)	utf8_general_ci	NO	MUL			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	collation	Cardinality	Sub part	Packed	Null	Index type	Comment
go2prodom	1	Index_go	1	go_num	A	2197	(NULL)	(NULL)		BTREE	
go2prodom	1	Index_prodom	1	prodom_num	A	2197	(NULL)	(NULL)		BTREE	

**Foreign Key Relationships**

FK Id	Reference Table	Source Column	Target column	Extra Info
FK_gomap_Go_go2prodom	go	`go_num`	`go_num`	ON DELETE CASCADE ON UPDATE CASCADE,
FK_gomap_Prodom_go2prodom	prodom	`prodom_num`	`prodom_num`	ON DELETE CASCADE ON UPDATE CASCADE

**go2prosite****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
go_num	varchar(50)	utf8_general_ci	NO	MUL			select,insert,update,references	
prosite_num	varchar(50)	utf8_general_ci	NO	MUL			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
go2prosite	1	Index_go	1	go_num	A	1175	(NULL)	(NULL)		BTREE	
go2prosite	1	Index_prosite	1	prosite_num	A	3527	(NULL)	(NULL)		BTREE	

**Foreign Key Relationships**

FK Id	Reference Table	Source Column	Target Column	Extra Info
FK_gomap_Go_go2prosite	go	`go_num`	`go_num`	ON DELETE CASCADE ON UPDATE CASCADE.
FK_gomap_Prosite_go2prosite	prosite	`prosite_num`	`prosite_num`	ON DELETE CASCADE ON UPDATE CASCADE

**pfam****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
pfam_num	varchar(7)	utf8_general_ci	NO	PRI			select,insert,update,references	
pfam_desc	varchar(100)	utf8_general_ci	NO	MUL			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
pfam	0	PRIMARY	1	pfam_num	A	8240	(NULL)	(NULL)		BTREE	
pfam	1	Index_desc	1	pfam_desc	A	8240	(NULL)	(NULL)		BTREE	

**pfam2cog****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
pfam_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
cog_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
pfam2cog	0	PRIMARY	1	pfam_num	A	5597	(NULL)	(NULL)		BTREE	
pfam2cog	0	PRIMARY	2	cog_num	A	5597	(NULL)	(NULL)		BTREE	

**pfam2prosite****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
pfam_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
prosite_num	varchar(7)	latin1_swedish_ci	NO	PRI			select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
pfam2prosite	0	PRIMARY	1	pfam_num	A	18082	(NULL)	(NULL)		BTREE	
pfam2prosite	0	PRIMARY	2	prosite_num	A	18082	(NULL)	(NULL)		BTREE	

**prodom****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
prodom_num	varchar(50)	utf8_general_ci	NO	PRI			select,insert,update,references	
prodom_desc	varchar(50)	utf8_general_ci	YES	MUL	(NULL)		select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
prodom	0	PRIMARY	1	prodom_num	A	761	(NULL)	(NULL)		BTREE	
prodom	1	Index_desc	1	prodom_desc	A	761	(NULL)	(NULL)	YES	BTREE	

**prosite****Fields**

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
prosite_num	varchar(7)	utf8_general_ci	NO	PRI			select,insert,update,references	
prosite_desc	varchar(50)	utf8_general_ci	YES	MUL	(NULL)		select,insert,update,references	

**Indexes**

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
prosite	0	PRIMARY	1	prosite_num	A	1677	(NULL)	(NULL)		BTREE	
prosite	1	Index_desc	1	prosite_desc	A	1677	(NULL)	(NULL)	YES	BTREE	

## Bibliography

## Bibliography

1. Yu, C., Zavaljevski, N., Desai, V., *Protein Function Prediction: Integrated Software Development Plan*. February 2006.
2. Machalek, A.Z., *From Genes to Proteins: NIGMS Catalogs the Shapes of Life*. NIH Record, 2001. **February**.
3. Whisstock, J.C. and A.M. Lesk, *Prediction of protein function from protein sequence and structure*. Q Rev Biophys, 2003. **36**(3): p. 307-40.
4. Tian, W., A.K. Arakaki, and J. Skolnick, *EFICAz: a comprehensive approach for accurate genome-scale enzyme function inference*. Nucleic Acids Res, 2004. **32**(21): p. 6226-39.
5. Claudel-Renard, C., et al., *Enzyme-specific profiles for genome annotation: PRLAM*. Nucleic Acids Res, 2003. **31**(22): p. 6633-9.
6. Porter, C.T., G.J. Bartlett, and J.M. Thornton, *The Catalytic Site Atlas: a resource of catalytic sites and residues identified in enzymes using structural data*. Nucleic Acids Res, 2004. **32**(Database issue): p. D129-33.
7. Pegg, S.C., et al., *Leveraging enzyme structure-function relationships for functional inference and experimental design: the structure-function linkage database*. Biochemistry, 2006. **45**(8): p. 2545-55.
8. Ashburner, M., et al., *Gene ontology: tool for the unification of biology. The Gene Ontology Consortium*. Nat Genet, 2000. **25**(1): p. 25-9.
9. Tatusov, R.L., et al., *The COG database: an updated version includes eukaryotes*. BMC Bioinformatics, 2003. **4**: p. 41.
10. Marchler-Bauer, A., et al., *CDD: a Conserved Domain Database for protein classification*. Nucleic Acids Res, 2005. **33**(Database issue): p. D192-6.
11. Durbin, R., Eddy, S., Krogh, A., and Mitchison, G: *Biological Sequence Analysis: probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, UK, 1998.
12. Chenna, R., et al., *Multiple sequence alignment with the Clustal series of programs*. Nucleic Acids Res, 2003. **31**(13): p. 3497-500.
13. Bairoch, A., *The ENZYME database in 2000*. Nucleic Acids Res, 2000. **28**(1): p. 304-5.
14. Ashburner, M., et al., *Gene ontology: tool for the unification of biology. The Gene Ontology*

- Consortium*. Nat Genet, 2000. **25**(1): p. 25-9.
15. Finn, R.D., et al., *Pfam: clans, web tools and services*. Nucleic Acids Res, 2006. **34**(Database issue): p. D247-51.
  16. Hulo, N., et al., *The PROSITE database*. Nucleic Acids Res, 2006. **34**(Database issue): p. D227-30.
  17. Servant, F., et al., *ProDom: automated clustering of homologous domains*. Brief Bioinform, 2002. **3**(3): p. 246-51.
  18. Webb, E.C., ed. *Enzyme Nomenclature*. 1992.
  19. Wikipedia contributors, *Screen scraping*. 28 November 2006 09:27 UTC [cited 30 November 2006 08:40 UTC]; 90628787 [Available from: [http://en.wikipedia.org/w/index.php?title=Screen\\_scraping&oldid=90628787](http://en.wikipedia.org/w/index.php?title=Screen_scraping&oldid=90628787)].
  20. Wikipedia contributors, *Web scraping*. 16 November 2006 01:28 UTC [cited 30 November 2006 08:47 UTC]; 88112702:[Available from: [http://en.wikipedia.org/w/index.php?title=Web\\_scraping&oldid=88112702](http://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=88112702)].
  21. George, R.A., et al., *Effective function annotation through catalytic residue conservation*. Proc Natl Acad Sci U S A, 2005. **102**(35): p. 12299-304.

## Curriculum Vitae

Seth Johnson was born Vsevolod Vladimirovich Ivanov on November 18, 1974, in Minsk, USSR, now Belarus. At the age of 5 his family moved to Vilnius, USSR, now Lithuania. In 1989, shortly before dismantlement of the USSR, he immigrated to the United States as a political refugee. He became an American citizen in 2000. He graduated from Samuel Wolfson Senior High School, Jacksonville, Florida, in 1993. He received his Bachelor of Science in Microbiology & Cell Science from University of Florida in 1998, Bachelor of Arts in Russian from University of Florida in 1998, and Bachelor of Science in Computer Science from University of Florida in 2001. He was employed as a scientific software developer in Florida, Maryland, and Virginia for almost five years. He currently holds a position of Senior Bioinformatics Associate with ExonHit Therapeutics, Inc. in Gaithersburg, MD charged with developing alternative splicing identification pipeline and database.