RECEIVED
NOV 08 1999
OSTI

FINAL REPORT

DOE SBIR Phase II

Application Number 36037-95-II

Grant Number DE-FG02-95ER81923

We have no objection from a patent
attorney to the publication of
this report.

11/4/99
Date
Patent & Intellectual
Property Counsel
New Field Office, Chicago

GRAIL-genQuest: A Comprehensive Computational System for DNA Sequence Analysis

For more information, contact:

Ruth Ann Manning, Ph.D.
Principal Investigator
ApoCom Inc.

11020 Solway School Road, Suite 101
Knoxville, TN 37931-2052

PHONE: 423.927.6120 FAX: 423.927.6122

EMAIL: apocom@apocom.com URL: <http://www.apocom.com>

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Project Summary: *Application Number 36037-95-II*

GRAIL-genQuest: A Comprehensive Computational System for DNA Sequence Analysis

Recent advances in DNA sequencing and genome mapping technologies are making it possible, for the first time in history, to find genes in plants and animals and to elucidate their function. This means that diagnostics and therapeutics can be developed for human diseases such as cancer, obesity, hypertension, and cardiovascular problems. Crop and animal strains can be developed that are hardier, resistant to diseases, and produce higher yields.

The challenge is to develop tools that will find the nucleotides in the DNA of a living organism that comprise a particular gene. In the human genome alone it is estimated that only about 5% of the approximately 3 billion pairs of nucleotides code for some 100,000 human genes. In this search for nucleotides within a genome which are active in the actual coding of proteins, efficient tools to locate and identify their function can be of significant value to mankind.

Software tools such as ApoCom GRAIL^{TM1} have assisted in this search. It can be used to analyze genome information, to identify exons (coding regions) and to construct gene models. Using a neural network approach, this software can "learn" sequence patterns and refine its ability to recognize a pattern as it is exposed to more and more examples of it.

Since 1992 versions of GRAILTM have been publicly available over the Internet from Oak Ridge National Laboratory. Because of the potential for security and patent compromise, these Internet versions are not available to many researchers in pharmaceutical and biotechnology companies who cannot send proprietary sequences past their data-secure firewalls. ApoCom is making available commercial versions of the GRAILTM software to run self-contained over local area networks.

As part of the commercialization effort, ApoCom has developed a new Java^{TM2}-based graphical user interface, the ApoCom Client Tool for Genomics (ACTG)TM. Two products, ApoCom GRAILTM Network Edition and ApoCom GRAILTM Personal Edition, have been developed to reach two diverse niche markets in the Phase III commercialization of this software. As a result of this project, ApoCom GRAILTM can now be made available to the desktop (UNIX®, Windows® 95 and Windows NT®, or MacTM OS)³ of any researcher who needs it.

¹ GRAIL is a trademark of the Lockheed Martin Energy Research Corporation. ApoCom GRAILTM is a commercial version of the Department of Energy developed GRAILTM Gene Characterization Software licensed to ApoCom by Lockheed Martin Energy Research Corporation.

² Java is a trademark of Sun Microsystems, Inc. in the United States and other countries.

³ UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. Windows 95 and Windows NT are registered trademarks of Microsoft Corporation in the United States and other countries. NT is a trademark of Northern Telecom Limited. Mac OS is a trademark of Apple Computer, Inc. in the United States and other countries.

APOCOM INC.

FINAL REPORT

**GRAIL-genQuest: A Comprehensive
Computational System
for DNA Sequence Analysis**

ACKNOWLEDGMENTS

This work was supported by DOE GRANT NO. DE-FG02-95ER81923.A003.

© 1998 ApoCom Inc. All Rights Reserved.

No part of this documentation may be photocopied, reproduced, or translated to another language without the prior written consent of ApoCom Inc.

ApoCom™ Client Tool for Genomics (ACTG) is a copyrighted work of ApoCom Inc.
© 1996 ApoCom Inc. All rights reserved.

ApoCom GRAIL™ Batch Tool (AGBT) is a copyrighted work of ApoCom Inc.
© 1997 ApoCom Inc. All rights reserved.

ApoCom GRAIL™ is a commercial version of the Department of Energy developed GRAIL™ Gene Characterization Software licensed to ApoCom Inc. by Lockheed Martin Energy Research Corporation.

ApoCom is a trademark of ApoCom Inc. in the United States.

GRAIL is a trademark of Lockheed Martin Energy Research Corporation.

Java and all Java-based trademarks and logos, including the Java Coffee Cup logo, are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

The Graphics Interchange Format is the copyright property of CompuServe Incorporated. GIF is a service mark property of CompuServe Incorporated. The LZW technology that is incorporated into the GIF is licensed under US Patent No. 4,558,302 and foreign counterparts from UNISYS CORPORATION.

Windows, Visual C++, Windows 95, and Windows NT are registered trademarks of Microsoft Corporation in the United States and other countries. NT is a trademark of Northern Telecom Limited.

All other product names are trademarks or registered trademarks of their respective companies.

Ruth Ann Manning, Ph.D.
Principal Investigator
ApoCom Inc.
11020 Solway School Road, Suite 101
Knoxville, TN 37931-2052
PHONE: 423.927.6120 FAX: 423.927.6122
EMAIL: apocom@apocom.com URL: <http://www.apocom.com>

TABLE OF CONTENTS

	page
1.0 INTRODUCTION	1
2.0 Key Issues/Scope	2
2.1 ApoCom GRAIL™ Made Available to Desktop of ALL Users.....	2
2.1.1 ApoCom Develops Seven Commercializable Products.....	2
2.1.2 Java™ Chosen for Graphical User Interface (GUI) Development	3
2.1.3 ApoCom Provides Client/Server Software for Proprietary Sequencing	3
2.2 ApoCom Develops International Market Base	5
2.3 ApoCom Exhibits GRAIL™ At Eight Trade Shows	5
2.4 ApoCom Assembles Technical Advisory Board of Leading Experts.....	5
2.5 ApoCom Meets All Objectives and Goals.....	7
3.0 TECHNICAL BACKGROUND.....	9
3.1 Where Does ApoCom GRAIL™ Fit In the Genomics Picture?	9
3.1.1 The Overall Picture.....	9
3.1.2 The Search for Disease Genes	10
3.1.3 DNA Collection.....	10
3.1.4 Genetic Linkage Maps.....	11
3.1.5 Physical Mapping	11
3.1.6 DNA Sequencing and Mutation Analysis.....	12
3.1.7 Diagnostics and Therapeutics Development.....	12
3.2 What is Java™?	13
3.3 What is Fuzzy Logic?	15
4.0 Work Performed	17
4.1 Objective 1 - Graphical User Interface Development.....	17
4.2 Objective 2 - Parallel Database Search Algorithms.....	175
4.3 Objective 3 - Frameshift Detection Algorithms.....	188
4.4 Objective 4 - User's Group and Technical Advisory Board.....	196
4.5 Objective 5 - New Strategies	205
4.6 Objective 6 - GRAIL™ Improvements	206
4.6.1 ACTG Window.....	206
4.6.2 DNA SEQUENCE Window	210
4.6.3 ANALYSIS Window.....	212
4.6.4 FEATURES Window	214
4.6.5 The ACTG Window Menus.....	215
4.6.5.1 File Menu.....	215
4.6.5.2 Grail 1-1a-2 Algorithm Menu.....	216
4.6.5.3 Features Menu	217
4.6.5.4 Window Menu	220
4.6.5.5 Assemble Menu	224
4.6.5.6 Zoom Slider and Scrollbar	226
4.6.5.7 Exit Option	227

1.0 INTRODUCTION

Vice President Al Gore states that "worldwide annual sales of biotechnology-derived products went from zero in 1980 to \$5.9 billion in 1992 and are expected to reach \$50 billion by the end of the century."⁴ The world of biotechnology is one of the fastest growing high tech industries in the world. It is not surprising that Vice President Al Gore recently acknowledged the emerging industry of biotechnology as the most exciting and fastest growing industry on into the next century. The Human Genome Project (HGP) and advances in sequencing and mapping technologies are spawning a revolution in the fields of medicine and biotechnology. Within the last few years alone, several dozen human disease genes have been found leading to billions of dollars worth of diagnostic and therapeutic products. "Recent headlines on single genes that cause rare inherited diseases will pale beside tomorrow's on patterns of genes predisposing us to heart attacks, cancer, or Alzheimer's disease.... Shock waves from such discoveries will rock the practice of medicine. No corner of health care, a nearly trillion dollar industry in the U.S. alone, will remain untouched. Most dramatic will be the impact on the \$180 billion-a-year worldwide pharmaceutical business."⁵

The number of biotechnology and pharmaceutical companies sequencing DNA in search of some 5,000 known human disease genes is increasing dramatically as technologies for sequencing become more efficient and cost-effective. While DNA sequencing in the Human Genome Project toward the 2005 target for sequencing the entire genome is occurring fairly systematically, biotechnology and pharmaceutical companies have focused on sequencing regions thought to contain particular disease genes. This is done in order to discover and characterize particular genes before they become public information, obtain necessary patent protection, and develop proprietary diagnostic and therapeutic technologies. This mode of operation (where sequencing is done in-house and computing is protected by firewalls) protects the company, but creates a need for secure, in-house computer analysis technologies such as ApoCom GRAILTM⁶ to provide the necessary knowledge on which to base the design of diagnostic and therapeutic products.

ApoCom GRAILTM is a gene and coding recognition system using multiple-sensor neural network systems for feature recognition. Signals and features recognized include exons or coding regions, splice sites, RNA Pol II promoters, cDNA coding regions, repetitive elements, CpG islands, polyadenulation sites, transcription and translation start and stop, and translations of coding regions and gene models to protein. ApoCom GRAILTM bridges the gap between proprietary mapping and sequencing and the development of DNA diagnostic and therapeutic products by providing a comprehensive computational framework for sequence analysis in medicine and biotechnology.

⁴ Gore, Al. *Government-Industry Cooperation Can Help U.S. Biotech Succeed in Global Marketplace in Genetic Engineering News*, Mary Ann Liebert, Inc. publishers, New York. 14(1), 1994, pp. 4, 12.

⁵ *The Biotech Revolution* in **FORTUNE**, March 31, 1997.

⁶ GRAIL is a trademark of the Lockheed Martin Energy Research Corporation. ApoCom GRAILTM is a commercial version of the Department of Energy developed GRAILTM Gene Characterization Software licensed to ApoCom Inc. by Lockheed Martin Energy Research Corporation.

2.0 Key Issues/Scope

2.1 ApoCom GRAIL™ Made Available to Desktop of ALL Users

The GRAIL™ Gene Characterization Software licensed to ApoCom by Lockheed Martin Energy Research Corporation was developed only for architectures developed by SUN Microsystems, Inc. This licensed software did not have a Graphical User Interface (GUI). This grant has enabled ApoCom to develop versions of ApoCom GRAIL™ for all major architectures (PC, Macintosh, SGI, SUN, DEC, IBM, Hewlett Packard, etc.) and all major operating systems (UNIX®, Windows® 95 and Windows NT®, and Mac™ OS). ApoCom chose the Java™ software development paradigm to develop both applet and application versions of its new copyrighted GUI, the ApoCom™ Client Tool for Genomics (ACTG). Altogether, ApoCom developed seven commercializable products for diverse niche biotechnology and pharmaceutical markets. These markets range from multidisciplinary teams at large biotechnology and pharmaceutical firms who are doing proprietary sequencing to much smaller research and development laboratories with limited budgets who are satisfied with the ever-increasing speeds of the PC on their desktop.

2.1.1 ApoCom Develops Seven Commercializable Products

Previously, GRAIL™ was only available to those researchers who could send their sequences out over the Internet to access the Oak Ridge National Laboratory online server software. Now, because of this DOE SBIR grant, ApoCom GRAIL™ is literally available to any scientist who needs it—from the researcher doing proprietary sequencing to the scientist with a Macintosh or PC on their desk. ApoCom has been successful in commercializing the following seven products:

- ApoCom Client Tool for Genomics (ACTG)™—a multi-platform client/server Java™ graphical user interface which displays many regions of interest including pol II promoters, repetitives, CpG eslands, polyA sites, protein translations and gene models.
- ApoCom GRAIL™—a commercial server version of the Department of Energy developed GRAIL™ Gene Characterization Software licensed to ApoCom by Lockheed Martin Energy Research Corporation (LMER) in 1994. GRAIL is a trademark of LMER.
- ApoCom GRAIL™ Network Edition—a multi-platform client/server edition which allows the combination of ALL resources (UNIX®, Windows® 95 and Windows NT®, and Mac™ OS) for genomic, cDNA, and EST sequence exploration and discovery.

- ApoCom GRAIL™ Personal Edition—a single user, single PC Java™ edition which unleashes the power of the desktop to analyze sequences both interactively and by automated computation.
- ApoCom GRAIL™ Batch Tool—a Visual C++ batch interface for the personal PC for initiating the automated analysis of over 30,000 sequences with a single request.
- ApoCom GRAIL™ Command Line Client for Network Edition—a sockets-based command-line interface for easy integration into existing bioinformatics analysis pipelines.
- BATCH ApoCom GRAIL™ for Network Edition—a client/server implementation for automated sequence analysis.

2.1.2 Java™ Chosen for Graphical User Interface (GUI) Development

"The Java™ platform is a fundamentally new way of computing, based on the power of networks and the idea that the same software should run on many different kinds of computers, consumer gadgets, and other devices. With Java technology, you can use the same application from any kind of machine -- a PC, a Macintosh computer, a network computer, or even new technologies like Internet screen phones."⁷ ApoCom had the foresight to recognize that this new technology would be the most exciting cross-platform tool that it has become since its inception three years ago. Even IBM recognized ApoCom's ingenuity in January, 1998 when a link was established from its WorldWide Web Home Page (<http://www.ibm.com/java>) to ApoCom's online applet demo (<http://www.apocom.com>) which IBM titled "Jump to Java Genetics."

2.1.3 ApoCom Provides Client/Server Software for Proprietary Sequencing

ApoCom is the only provider of the GRAIL™ software to run in-house over local area networks or on the desktop. ApoCom has made this state-of-the-art software accessible to the researcher doing proprietary sequencing who cannot send sequences out over the Internet. ApoCom GRAIL™ is a suite of tools designed to provide analysis of DNA sequences both interactively and by automated computation. The GRAIL™ Gene Characterization Software is a DNA Sequence Analysis and Gene Localization System developed at Oak Ridge National Laboratory based on statistical, syntactic, and neural network methods. This software recognizes and identifies putative exon coding regions within user-supplied DNA sequences and supports a wide range of analysis including gene modeling.

⁷ <http://java.sun.com/nav/whatis/>

Features Recognized By ApoCom GRAIL™

Coding exons

- preferred reading frame
- 3' and 5' splice sites
- protein translation

Pol II promoters

- TATA box
- CAAT box
- GC box
- Cap site

Coding strand determination

CpG islands

Gene models

Gene model translation

Local GC content

Polyadenylation sites

Repetitive elements

The ApoCom GRAIL™ neural network system currently has three (3) analysis options (**Grail 1**⁸, **Grail 1a**, and **Grail 2**) with capabilities appropriate for different types of sequencing projects. **Grail 1** (and the newer **Grail 1a**) are designed to provide analysis when cDNAs, ESTs, or other isolated fragments of sequence (without genomic context) are the targets. The **Grail 1** systems rely primarily on neural network-based coding prediction information to recognize coding portions of cDNAs, ESTs, trapped exons, or other isolated fragments of a sequence (without genomic context). The **Grail 1** systems also recognize pseudo-genes in which edge signals (splice sites, etc.) are possibly corrupted.

Grail 2 is designed for the analysis of genomic regions where the goal is to recognize and characterize exons with intact splice, start, or stop edge signals and to construct a gene model. **Grail 2** uses additional genomic context information to improve the accuracy and sensitivity of the exon recognition process (*e.g.*, splice junctions, translation starts, non-coding scores of 60-base regions on either side of a putative exon). **Grail 2** coding predictions inherently contain specific edge signals and can be used as the basis for gene modeling. Heuristic methods and dynamic programming are used in the gene assembly program (GAP) to predict the spliced mRNA message from the genomic DNA sequence.

The ApoCom GRAIL™ software facilitates an efficient and accurate gene model construction capability based on interactive user input and **Grail 2** analysis of coding regions, translation start sites, and other signals. Additional signals recognized include CpG islands, RNA Pol II promoters, repetitive elements (*e.g.*, Alu), polyadenylation sites, and translations of coding regions and gene models to protein. The **Grail 1**, **Grail 1a** and **Grail 2** systems provide very accurate prediction of the translation frame of a recognized coding region and protein translation in the "preferred" frame.

⁸ Uberbacher EC, Mural RJ. *Locating protein coding regions in human DNA sequences by a multiple sensor neural network approach*. Proc. of National Academy of Sciences USA 1991; 88: 11261-11265.

2.2 ApoCom Develops International Market Base

ApoCom has developed an international market base for its products. In addition to pharmaceutical and biotechnology companies in the United States, ApoCom has customers in the United Kingdom, Sweden, Germany, France, and Japan.

2.3 ApoCom Exhibits GRAIL™ At Eight Trade Shows

ApoCom has exhibited ApoCom GRAIL™ at eight (8) trade shows from 1994-present. These trade shows include Human Genome 1994: The Genes and Beyond, several TIGR Genome Sequencing and Analysis Conferences, and Supercomputing 1994. ApoCom has also maintained a dynamic WWW Home Page and advertised in *Microbial & Comparative Genomics* and *The Biotechnology Software Journal*. ApoCom's products have also been featured in *Genetic Engineering News* and *bioinform*.

2.4 ApoCom Assembles Technical Advisory Board of Leading Experts

ApoCom assembled a 9-member Technical Advisory Board consisting of leading experts from the US and Europe in the fields of DNA sequence analysis research, pharmaceutical therapeutics and diagnostics development, and biotechnology business leaders.

ApoCom Technical Advisory Board

- **Lydie Bougueleret, Ph.D.** **GENSET SA, France**

GENSET is a public biotechnology company engaged in the systematic and comprehensive analysis of the human genome. Their mission is to identify and patent genes and regulatory regions related to selected common diseases (including prostate cancer, schizophrenia, and osteoporosis), to discover drugs for such diseases, and to enter into strategic partnerships with pharmaceutical companies which will develop and market these drugs. GENSET has developed the NetGene database and Biointelligence analysis software.

- **Richard Gibbs, Ph.D.** **Baylor College of Medicine**

Richard is the **Director** of the **Baylor Human Genome Sequencing Center**. The research from his laboratory is focused in three main areas: 1) polymerase chain reaction (PCR) methods for studying the molecular basis of human X-linked genetic diseases; 2) the molecular basis of somatic mutations in mammalian cells; and 3) direct sequencing of the human immunodeficiency virus (HIV).

- **Jim Golden, Ph.D.**

GeneSys Inc.

Jim is recognized as a leading expert in the field of neural networks and DNA sequence analysis software development. He also consults for ApoCom on our SBIR Phase II Grant. GeneSys develops instruments for automated DNA sequencing.

- **Jim Lindelien**

Time Logic, Inc.

Jim is the founder and President of Time Logic. Time Logic's **DeCypher II** is a scalable, high-performance array processor hardware technology which can be added to Unix, Mac, and PC workstations to accelerate Smith-Waterman and other popular genetic similarity algorithms to speeds well in excess of present supercomputers.

- **Dick McCombie, Ph.D.**

Cold Spring Harbor Laboratory

Cold Spring Harbor Laboratory is an independent, nonprofit research and educational institution chartered by the University of the State of New York. When James Watson became the Director in 1969, he focused the Laboratory's research on the genetics of cancer, using viruses as a model system. Dick McCombie, Richard Gibbs, Ellson Chen (Perkin Elmer Corporation), and Richard Wilson (Washington University) team each year to teach a two-week course on recent advances in high speed automated DNA sequencing. The Cold Spring Harbor *Arabidopsis* Genome Sequencing Project uses genetic approaches coupled with the high throughput sequencing of genomic DNA and cDNA tags from selected libraries. Cold Spring Harbor is also committed to the characterization of the fission yeast *Schizosaccharomyces pombe* genome.

- **Yannick Pouliot, Ph.D.**

Molecular Simulations, Inc.

Molecular Simulations provides computational chemistry software and services to organize scientific data, share information, and develop, analyze, and simulate novel compounds and processes. Specific tools are provided for analytical instrumentation, catalysis and sorption, chemical reactions, crystallization, drug design, molecular biology, NMR, polymer modeling, protein design, quantum mechanics, software development, and X-ray crystallography. Yannick is now with Progenitor.

- **Bruce Roe, Ph.D.**

University of Oklahoma

Bruce is the Director of the Advanced Center for Genome Technology, located on the Norman Campus of the University of Oklahoma. This Center is a state-of-the-art DNA sequencing facility, having completed the sequence of greater than three million bases of human genomic DNA from cosmid, P1 and BAC clones from human chromosome 22.

- **Eric Snyder, Ph.D.**

**AxyS Pharmaceuticals (formerly
Sequana Therapeutics Inc.)**

Eric and his major professor G.D. Stormo (University of Colorado) developed a well-known neural networks software package called **GeneParser** which is similar to **GRAIL™ GeneParser** identifies and determines the fine structure of protein genes in genomic DNA sequences. Eric is also a consultant to ApoCom for the SBIR Phase II Grant. Sequana Therapeutics is searching for disease genes, including type II diabetes, asthma, osteoporosis and schizophrenia. The identification of disease genes will provide insights into the causes of common diseases and will facilitate the development of highly specific diagnostic and therapeutic products, such as small molecule drugs, recombinant proteins, gene therapy and antisense therapy.

- **Clark Tibbetts, Ph.D.**

Vanderbilt University

Clark is a Professor of Microbiology at Vanderbilt University. He is a recognized leader in the field of neural networks. Clark has developed a contextual basecalling system which operates as a pipeline of signal conditioning algorithms and neural network data stream transformations. His research in the area of high speed DNA sequencers using arrays of microchannels etched in a glass substrate is leading the edge of high throughput DNA sequencing technologies.

2.5 ApoCom Meets All Objectives and Goals

All of the following primary goals of this project have been met:

- Develop cross-platform client graphical user interfaces (GUIs) for Mac™ OS, Windows® 95, Windows NT®, and UNIX operating systems based on design input from both Internet and commercial GRAIL™-genQuest users. This objective addresses the need for easy-to-use centralized server software that can be accessed company wide by non-specialists using client tools on the desktop PC's, Macintoshes, or workstations.
- Develop both shared- and distributed-memory parallel implementations of the Smith-Waterman-Gotoh database search and comparison algorithms which will improve response time for searches and increase user capacity per server (with a goal of less dependence on specific architectures).
- Develop and parallelize new database search algorithms for DNA-protein sequence comparison which detect multiple frameshifts in the DNA that effect translation. These algorithms can recognize very weak homologies between a translated DNA with multiple frameshifts and the SwissProt protein database with an accuracy and sensitivity of Smith-Waterman (at several times the computational cost). These

algorithms will be especially valuable to customers who are doing single pass sequencing.

- Form a GRAIL™-genQuest User's Group and a Technical Advisory Board. Conduct comprehensive surveys of commercial and Internet users to improve the visualization capabilities, functionality and GUI design for the display of sequence-level features, multiple organism comparison, and other identified needs which can be implemented during Phase II.
- Develop new computer tools, information systems, and strategies for the annotation, management, analysis, interpretation, and dissemination of large amounts of genomic mapping and sequencing information..
- Improve GRAIL™'s technology for the identification of all coding sequences, genes, and other functional elements in genomic DNA (from both non-human organisms and the human).

The ORNL Gene Recognition and Internet Link (GRAIL™) has been the most widely used client/server Internet system for locating and characterizing genes in DNA sequences. ApoCom GRAIL™ can be used to analyze both short isolated sequence fragments and larger regions containing partial or multiple genes. This system formerly was not accessible to many researchers because their primary computers are Macintoshes and personal computers (PCs).

ApoCom provides a range of analysis and database search systems in support of genomic, cDNA, and EST sequencing. ApoCom GRAIL™ features a Java-based client graphical interface, ApoCom™ Client Tool for Genomics (ACTG), and a sockets-based command line interface for easy integration into existing user environments. Both client and server software for ApoCom GRAIL™ run on the users' own local area networks.

ApoCom has been successful in commercializing seven (7) products. The ApoCom GRAIL™ Network Edition enables pharmaceutical and biotechnology companies to use the ApoCom commercial suite of analysis tools on a variety of platforms. The ApoCom™ Client Tool for Genomics (ACTG) is a Java™-based visual exploration tool for DNA sequence analysis and discovery. ACTG allows the user to interactively explore and display many regions of interest including RNA Pol II promoters, repetitives, CpG islands, protein translations, and gene models. The ApoCom Client Tool for Genomics (ACTG)™ can run either as a standalone application on platforms which can run a Java™ interpreter (e.g., Windows 95, Windows NT, Apple Macintosh, SGI IRIX and SUN Solaris 2.x). ACTG is also readable as an applet by any Java™-compatible World-Wide Web browser.

The ApoCom GRAIL™ Batch Tool (AGBT) is a Microsoft® Visual C++ graphical user interface which provides an alternative batch interface to all of the same analysis options available in ACTG. With the batch tool you can initiate the automated analysis of over 30,000 sequences with a single request.

The ApoCom GRAIL™ Personal Edition is a special single-user, single-PC version which combines ACTG and AGBT to unleash the power of desktop computers to analyze sequences both interactively and by automated computation.

3.0 TECHNICAL BACKGROUND

3.1 *Where Does ApoCom GRAIL™ Fit In the Genomics Picture?*

3.1.1 The Overall Picture

By the end of the nineteenth century biologists had recognized that the **chromosomes** that become visible in the nucleus as a cell begins to divide were the carriers of inherited information. But the evidence that the **deoxyribonucleic acid (DNA)** in these chromosomes is the substance of which genes are made was not generally accepted until the early 1950s. The human genetic code (**genome**) consists of 23 pairs of chromosomes contained in the nucleus of nearly every living cell in the body, one member of each pair inherited from each parent.

A DNA **nucleotide** is a sub-unit of DNA consisting of a nitrogenous base (adenine, guanine, thymine or cytosine, represented by the letters A, G, T, C), a phosphate molecule, and a sugar molecule (deoxyribose). Two strands of DNA are held together by the bonds between **base pairs (bp)**—two nucleotides (A and T or G and C) held together by weak bonds. These strands of a DNA molecule wrap around each other to form a right-handed **double helix** with about 10 nucleotide pairs per helical turn. This structure resembles a twisted ladder whose “sides,” made of sugar and phosphate molecules, are connected by “rungs” of nitrogen-containing pairs of bases. The particular order of the bases arranged along the sugar-phosphate backbone of a DNA strand is called the **DNA sequence**.

Ribonucleic acid (RNA) resembles DNA having a sugar-phosphate backbone with a series of bases attached. The differences are that the sugar of RNA is ribose, it is usually single-stranded, and RNA is much shorter (usually 50-1000 nucleotides). The bases A, C, and G are the same as in DNA, but the base thymine (T) is replaced by another base called uracil (U) which, like thymine, is complementary to adenine (A).

A **gene** is an ordered sequence of nucleotides located in a particular position on a particular chromosome that has enough information in it to specify a particular **protein**. Human genes vary widely in length, often extending over thousands of bases. Proteins are large, complex molecules made up of chains of **amino acids**. There are 20 amino acids which make up a protein chain. Within a gene, a sequence of three DNA bases (**codons**) directs a cell to produce a specific amino acid. For example, the sequence AUG codes for the amino acid methionine. So the protein coded by an average-sized human

gene (3000 bp) will contain 1000 amino acids. The human genome is estimated to contain about 100,000 genes.

It is now a well-established fact that proteins are the workhorse molecules of life. They represent the engines that drive life processes from metabolic activities and cell synthesis to combating disease and degrading foreign compounds. A common thread tying all living things together is the subcellular synthesis of proteins comprised of amino acids whose creation is dictated by the sequence of nucleotides within DNA. The sequence of nucleotide bases within DNA therefore act as the templates for peptide and amino acid synthesis and subsequently for protein synthesis. In the human genome alone, there are a total of 3 billion nucleotide pairs, and less than 5% of these are known to be employed in the actual coding of proteins. Thus it is useful to search for patterns of DNA that mark the boundaries between coding (**exon**) regions and noncoding (**intron**) regions.

3.1.2 The Search for Disease Genes

Although the basic information in the genome of each human is similar, each individual has a unique DNA “fingerprint.” DNA **polymorphisms** are natural variations in DNA sequence that differ from individual to individual and are passed on from generation to generation. Polymorphisms occur on the average of once every 300-500 bp. Alterations or **mutations** in genes can result in improper gene product production. If the affected protein plays a critical role (e.g., an enzyme, hormone, or receptor), the result is often **disease** or a predisposition toward disease. If a polymorphism is often inherited with a particular disease, it is often “linked” to particular chromosomal DNA regions containing mutations responsible for disease predisposition.

The task of finding a specific gene has been daunting. This complexity has been compared⁹ with that of trying to find a single burning house somewhere within the world’s 100,000 major cities without any further information about its regional location. Borrowing from this analogy, an approach to the discovery of disease genes (including the roles of ApoCom GRAIL™ and the ApoCom Client Tool for Genomics (ACTG)™¹⁰) will now be outlined.

3.1.3 DNA Collection

The first step in discovering new disease genes is the collection of DNA from large numbers of affected individuals and their families and from control populations. Usually automated, high-efficiency DNA processing systems prepare and label DNA from patient blood or animal samples. This might be compared to collecting data about house fires from the world’s 100,000 major cities.

⁹ Sequana Therapeutics Home Page URL: <http://www.sequana.com> now AxyS Pharmaceuticals.

¹⁰ ApoCom Client Tool for Genomics (ACTG)™ is a copyrighted work of ApoCom Inc. © 1996. All rights reserved.

3.1.4 Genetic Linkage Maps

Fluorescent genetic markers spaced at regular intervals throughout the genome can be used to label the DNA (**genotyping**). Boundaries for the location of potential **disease genes** can be defined by studying polymorphism patterns in DNA from affected and nonaffected individuals. It has been pointed out that labeling the genome with polymorphic markers is analogous to marking the earth's surface with latitude and longitude lines. In both cases the resolution can be increased by using more markers.

After genotyping, statistical methods can be used to correlate (or link) the inheritance of polymorphic markers with the incidence of a particular disease. A **genetic linkage map (genome map)** describes the order and position of genes and other DNA markers on each chromosome. Genome maps allow researchers to narrow the search for a disease gene to a particular region on the chromosome. This is analogous to determining that the burning house is somewhere in California.

3.1.5 Physical Mapping

Once the genetic interval containing a disease gene has been identified, some type of **physical map** is constructed to ascertain the approximate location of this gene. Different types of physical maps vary in their degree of resolution. The lowest resolution physical map is the **chromosomal (or cytogenetic) map**, based on the distinctive banding patterns observed by light microscopy of stained chromosomes. A **cDNA map** depicts the locations of expressed DNA regions (exons) on the chromosomal map. The more detailed **cosmid contig map** shows the order of overlapping DNA fragments spanning the genome. A commercially-available library of human genomic DNA fragments inserted into yeast cells to produce Yeast Artificial Chromosomes (YACs) can be overlapped to reconstruct the genetic region of interest. The highest resolution physical map will detail the complete DNA base pair sequence of each chromosome in the genome. The physical mapping process could be compared to getting a list of all major cities in the coastal counties of California and eliminating those major cities not in San Diego county.

ApoCom GRAIL™ can be used to analyze genotype information to identify exons and construct gene models. Using a neural network approach, ApoCom GRAIL™ can learn sequence patterns and refine its ability to recognize a pattern as it is exposed to more and more examples of it. By comparing data with existing DNA and protein databases for similarities (homology searches) researchers can eliminate some gene sequences while focusing on the genes most likely to be involved in the biochemical pathway of the disease. Similarities of unknown gene sequences to known genes can reveal similarity in function.

3.1.6 DNA Sequencing and Mutation Analysis

After identifying the approximate location of genes using physical mapping, genes are then sequenced from normal, healthy individuals for comparison with genes sequenced from individuals affected with the particular genetic disease. Both automatic sequence analyzers and bioinformatics (computational approaches and tools such as ApoCom GRAIL™) are used to compare sequences and identify differences between normal and affected genes. Sometimes major differences (such as DNA insertions or deletions) are found between the sequences of affected and normal individuals, but more often the differences are subtle, involving only a few nucleotides. This can be compared to using a helicopter to search for the burning house in San Diego county. A big fire spanning several blocks would be easier to locate than a smoldering fire in a single house.

3.1.7 Diagnostics and Therapeutics Development

The market for sequencing technology is as far reaching as the field of biotechnology itself. It includes the medical field and the synthesis of protein-based medicines and diagnostics related to hereditary diseases, as well as agricultural sciences, environmental bioremediation, and toxicology. By studying other model organisms such as mouse, *Arabidopsis*, *Drosophila*, yeast, and *E. coli* we can determine which genes are similar in function to other organisms (such as humans).

3.2 What is Java™?

"Nothing short of the World Wide Web itself has captured the attention and imagination of the Internet community as much as Java. What is Java? Java is two things: a programming language and a platform. Java is a high-level programming language similar to C/C++ that is simple, architecture-neutral, object-oriented, portable, distributed, high-performance, interpreted, multithreaded, robust, dynamic, and secure."¹¹

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called Java bytecodes--the platform-independent codes interpreted by the Java interpreter. With an interpreter, each Java bytecode instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

The most common Java programs are applications and applets. Applications are standalone programs, such as the HotJava browser. Applets are similar to applications, but they don't run standalone. Instead, applets adhere to a set of conventions that lets them run within a Java-compatible browser.

You can think of Java bytecodes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java bytecodes help make "write once, run anywhere" possible. You can compile your Java program into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, SUN Solaris, and Macintosh.

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components: the Java Virtual Machine (Java VM) and the Java Application Programming Interface (Java API). The Java VM is the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (packages) of related components.

¹¹Gosling, James and McGilton, Henry. *The Java Language Environment*. <http://java.sun.com/docs/white/platform/CreditsPage.doc.html>

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring Java's performance close to that of native code without threatening portability.

The most common types of programs are probably applets and applications, where a Java application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers include Web servers, proxy servers, mail servers, print servers, and boot servers. Another specialized program is a servlet. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java servers, configuring or tailoring the server.

How does the Java API support all of these kinds of programs? With packages of software components that provide a wide range of functionality. The core API is the API included in every full implementation of the Java platform. The core API gives you the following features:

- The Essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- Applets: The set of conventions used by Java applets.
- Networking: URLs, TCP and UDP sockets, and IP addresses.
- Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- Security: Both low-level and high-level, including electronic signatures, public/private key management, access control, and certificates.
- Software components: Known as JavaBeans, can plug into existing component architectures such as Microsoft's OLE/COM/Active-X architecture, OpenDoc, and Netscape's Live Connect.
- Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- Java Database Connectivity (JDBC): Provides uniform access to a wide range of relational databases.

Java not only has a core API, but also standard extensions. The standard extensions define APIs for 3D, servers, collaboration, telephony, speech, animation, and more.”¹²

¹² <http://java.sun.com/docs/books/tutorial/getStarted/index.html>

3.3 What is Fuzzy Logic?

Formal Definition of a Fuzzy System

We define a fuzzy system as a function

$$f : (A_0 \times \cdots \times A_{n-1}) \rightarrow (A_n \times \cdots \times A_{n+m-1})$$

where the A_k are sets. Let $i \in [0, n-1]$, $j \in [0, m-1]$, $k \in [0, n+m-1]$ be integer variables, and define $A_k^* = [0, 1]^{A_k} \cup \{*\}^{A_k}$. The function f is represented by the 4-tuple $\langle R, \wedge, \vee, d \rangle$ where:

- $R \subset (A_1^* \times \cdots \times A_{n+m}^*)$ is a finite nonempty set;
- $\wedge : ([0, 1] \cup \{*\})^n \rightarrow [0, 1]$ is a function;
- $\vee : 2^{A_{n+j}} \rightarrow [0, 1]^{A_{n+j}}$ is a polymorphic function; and
- $d : [0, 1]^{A_{n+j}} \rightarrow A_{n+j}$ is a polymorphic function.

Extend the minimum function so that $\min(*, y) = *$ for all $y \in [0, 1]$, and define $@ : (A_n^* \times \cdots \times A_{n+m-1}^*) \times [0, 1] \rightarrow (A_n^* \times \cdots \times A_{n+m-1}^*)$ where, using infix notation, $(q@y)_j(x) = \min(q_j(x), y)$. For all $r \in R$ let $r(a) = \langle r_0(a_0), \dots, r_{n-1}(a_{n-1}) \rangle$, and $\vec{r} = \langle r_n, \dots, r_{n+m-1} \rangle$.

The fuzzy system is defined by

$$f(a)_j = d \left(\vee \left\{ \left(\vec{r} @ \wedge (r(a)) \right)_j : r \in R \right\} \right)$$

Example Fuzzy System

The figure below illustrates the function computed by a simple fuzzy system represented by the following four rules:

If X is small and Y is small then Z is large

if X is large and Y is large then Z is large

If X is small and Y is large then Z is small

if X is large and Y is small then Z is small

Function Computed by a Simple Four-Rule Fuzzy System

4.0 Work Performed

ApoCom has been successful in meeting all of its objectives.

4.1 Objective 1 - Graphical User Interface Development

Develop cross-platform client graphical user interfaces (GUIs) for Mac™ OS, Windows® 95, Windows NT®, and UNIX operating systems based on design input from both Internet and commercial GRAIL-genQuest users. This objective addresses the need for easy-to-use centralized server software that can be accessed company wide by non-specialists using client tools on the desktop PC's, Macintoshes, or workstations.

ApoCom has developed both Java 1.0 and Java 1.1 versions of the ApoCom Client Tool for Genomics (ACTG). A listing of this code follows.

```

//////////
// ACTG.java
//////////

import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;

public class ACTG extends Applet
    implements ActionListener
{
    boolean amApplet = true;

    int userID;
    int fwdPort;
    int mainServerPort;
    String mainServerHostname;
    String fileServerHostname;

    TextField warningLabel; // Used only by the Applet
    TextField userIDTextField; // Used only by the Applet

    Grail control;

    //////////////////////////////////////
    // init ()
    //
    // init() is called each time the applet is loaded.
    // One can think of it as being the class constructor.
    //////////////////////////////////////
    public void init ()
    {
        // Make some stuff
        TextArea ta = new TextArea((new Copyright()).text,40,80);
        ta.setEditable(false);
        warningLabel = new TextField("");
        warningLabel.setEditable(false);
        userIDTextField = new TextField("",20);
        userIDTextField.addActionListener(this);
        userIDTextField.setEditable(true);
        Label userIDLabel = new Label("User ID Number :");
        setLayout(new BorderLayout());

```

```

this.add("Center",ta);

// Get the main server host name from the HTML page
mainServerHostname = getParameter("MAINSERVERHOSTNAME");
if ((mainServerHostname == null) ||
    (mainServerHostname.equals(""))) {
    warningLabel.setText(warningLabel.getText()+
        "Missing main server hostname - ");
}

// Get the file server host name from the HTML page
fileServerHostname = getParameter("FILESERVERHOSTNAME");
if ((fileServerHostname == null) ||
    (fileServerHostname.equals(""))) {
    warningLabel.setText(warningLabel.getText()+
        "Missing file server hostname - ");
}

// Get the main server port number from the HTML page
String mainServerPortString = getParameter("MAINSERVERPORT");
if (mainServerPortString == null) {
    warningLabel.setText(warningLabel.getText()+
        "Missing main server port -");
}
try {
    mainServerPort =
        Integer.valueOf(mainServerPortString).intValue();
    if (mainServerPort <= 0)
        warningLabel.setText(warningLabel.getText()+
            "Invalid main server port -");
}
catch (NumberFormatException e) {
    warningLabel.setText(warningLabel.getText()+
        "Invalid main server port -");
}

// Get the forwarder port number from the HTML page
String fwdPortString = getParameter("FORWARDERPORT");
if (fwdPortString == null) {
    warningLabel.setText(warningLabel.getText()+
        "Missing forwarder port -");
}
try {
    fwdPort =
        Integer.valueOf(fwdPortString).intValue();
    if (fwdPort <= 0)

```

```

        warningLabel.setText(warningLabel.getText()+
            "Invalid forwarder port -");
    }
    catch (NumberFormatException e) {
        warningLabel.setText(warningLabel.getText()+
            "Invalid forwarder port -");
    }

    // If the HTML page specifies valid parameters then
    // enable the text box so the user can type in their id number
    if (warningLabel.getText().equals("")) {
        Panel p = new Panel();
        p.add(userIDLabel);
        p.add(userIDTextField);
        this.add("South",p);
    }
    else this.add("South",warningLabel);
}

////////////////////////////////////
// start ()
//
// start() is called each time the page is revisited
////////////////////////////////////
public void start ()
{
    if (control != null) control.start();
}

////////////////////////////////////
// stop ()
//
// stop() is called each time the page is left. Since this method
// is empty, this applet will continue to run even when the browser
// leaves the web page that launched it.
////////////////////////////////////
public void stop ()
{
    if (control != null) control.stop();
}

////////////////////////////////////
// destroy ()

```

```

//
// destroy() is called each time the applet is unloaded
////////////////////////////////////
public void destroy ()
{
    if (control != null) control.destroy();
}

////////////////////////////////////
// dispose ()
//
// dispose() will be called by the control program when the user exits
////////////////////////////////////
void dispose ()
{
    control = null;
}

////////////////////////////////////
// main ()
//
// main() simply provides an execution environment
// in which to run the "applet" if a browser is not being used
////////////////////////////////////
public static void main (String args[])
{
    boolean nocopyright = false;
    String mainServerHost=null;
    int mainServerPort=0;
    int arg=0;

    // make a new "applet"
    ACTG app = new ACTG();

    // tell the applet it is running in application mode
    app.amApplet = false;

    // Parse the command line
    if ((args.length < 2) || (args.length > 3)) {
        System.out.print("Usage : java ACTG [-acceptcopyright] <server name>");
        System.out.println(" <server port number>");
        System.exit(1);
    }
}

```



```

// Check for the -nocopyright option
if (args[0].equals("-acceptcopyright")) {
    if (args.length != 3) {
        System.out.print("Usage : java ACTG [-acceptcopyright]");
        System.out.println(" <server name> <server port number>");
        System.exit(1);
    }
    nocopyright = true;
    arg++;
}

// Get the hostname
mainServerHost = args[arg];
arg++;

// Get the port number
try {
    mainServerPort = Integer.valueOf(args[arg]).intValue();
}
catch (NumberFormatException e) {
    System.out.println("Invalid <server port number> : "+args[arg]);
    System.exit(1);
}
if (mainServerPort <= 0) {
    System.out.println("Invalid <server port number> : "+args[arg]);
    System.exit(1);
}

// Make a new Grail object
app.control = new Grail(app, mainServerHost, mainServerPort,
    "none", 0, 0);

// Dump the copyright message to stdout if appropriate
if (!nocopyright) {
    System.out.print((new Copyright()).text);
}

// simulate the starting of the applet
app.start();

// Note that in application mode there is no equivalent of
// the applet stopping or being unloaded. Thus we do not need
// to call the stop() and destroy() methods
}

```

```

////////////////////////////////////
// handleEvent ()
//
// handleEvent() just waits for the user to type in his ID number
////////////////////////////////////

```

```

public void actionPerformed (ActionEvent evt)
{
    if (evt.getSource() == userIDTextField) {

        // Barf if they haven't entered anything yet
        if (userIDTextField.getText() == null) {
            return;
        }

        // Try to interpret what they type as an integer
        try {
            userID = Integer.valueOf(
                userIDTextField.getText()).intValue();
        }

        // Barf if they didn't type an integer
        catch (NumberFormatException e) {
            userIDTextField.selectAll();
            return;
        }

        // Barf if they entered a non-positive integer
        if (userID <= 0) {
            userIDTextField.selectAll();
            return;
        }

        // Everything OK so disable the text box and fire up
        // a new ACTG control

        // Tell them what's going on
        showStatus("Downloading class files");

        // Disable the text field
        userIDTextField.setEnabled(false);

        // Make the new Grail object
        control = new Grail(this,
            mainServerHostname, mainServerPort,

```

```
        fileServerHostname, userID, fwdPort);

    // Tell them we're firing it up
    showStatus("Starting ACTG");

    // Fire it up
    control.start();

    // Tell them it's active
    showStatus("ACTG is running");
}

return;
}
```

```
////////////////////
```

```
// AnalysisWindow.java
```

```
////////////////////
```

```
import java.awt.*;
import java.awt.event.*;
import java.awt.AWTEvent;
```

```
class AnalysisWindow extends ThreadedFrame
{
    final static int SETVERSION = -1;
    final static int SHOW_GENE_MODELS = -2;
    String grailExonLabel, modelExonLabel, geneModelLabel;
    Panel titlePanel, tablePanel, modelExonPanel;
    Grail myParent;
    Label title;

    MenuItem printButton, closeButton;
    Choice currentTable;

    final static String mainTitle = "ANALYSIS";
    final static String buttonName = "Analysis";

    AnalysisWindow (Grail g)
    {
        super(mainTitle + ": Sequence: None");

        // Store pointer to grail window
        myParent = g;

        // Set up the menus
        printButton = new MenuItem("Print");
        printButton.setEnabled(false);
        closeButton = new MenuItem("Close");
        closeButton.addActionListener(this);
        Menu fileMenu = new Menu("File");
        fileMenu.add(printButton);
        fileMenu.addSeparator();
        fileMenu.add(closeButton);
        MenuBar mb = new MenuBar();
        mb.add(fileMenu);
        setMenuBar(mb);
    }
}
```

```

// Set the layout manager of the top level window
setLayout(new BorderLayout());

// Create the choice menu
currentTable = new Choice();
currentTable.addItem(grailExonLabel="GRAIL EXON TABLE");
currentTable.addItem(modelExonLabel="MODEL EXON TABLE");
currentTable.addItem(geneModelLabel="GENE MODEL TABLE");
currentTable.addItemListener(this);

// Create the title panel
titlePanel = new Panel();
titlePanel.setLayout(new BorderLayout());
titlePanel.setFont(new Font("Courier",Font.PLAIN,14));
titlePanel.add("West",currentTable);
title = new Label();
titlePanel.add("East",title);
add("North",titlePanel);

// Create the table panel
tablePanel = new Panel();
tablePanel.setLayout(new CardLayout());
tablePanel.add(Grail.exons[0].button.getLabel(),
    Grail.exons[0].textDisplay);
tablePanel.add(Grail.exons[1].button.getLabel(),
    Grail.exons[1].textDisplay);
tablePanel.add(Grail.exons[2].button.getLabel(),
    Grail.exons[2].textDisplay);
modelExonPanel = new Panel();
modelExonPanel.setLayout(new BorderLayout());
tablePanel.add(modelExonLabel,modelExonPanel);
tablePanel.add(geneModelLabel,Grail.geneModels.textDisplay);
add("Center",tablePanel);

// Make sure the version is correctly set
setVersion();

// Call pack() to size the thing
pack();
}

void setVersion ()
{
    int v = Grail.ver;

```

```

    if (v != 2)
    {
        currentTable.select(grailExonLabel);
        currentTable.setEnabled(false);
    }
    else currentTable.setEnabled(true);

    if (currentTable.getSelectedItem().equals(grailExonLabel))
    {
        title.setText(Grail.exons[v].button.getLabel() +
            " # of Exons: " + Grail.exons[v].num);
        titlePanel.validate();
        ((CardLayout)tablePanel.getLayout()).
            show(tablePanel,Grail.exons[v].button.getLabel());
    }

    validate();
}

void showModels ()
{
    if (currentTable.getSelectedItem().equals(geneModelLabel))
    {
        title.setText("# of Models: " + Grail.geneModels.num);
        titlePanel.validate();
        ((CardLayout)tablePanel.getLayout()).
            show(tablePanel,geneModelLabel);
    }
}

void showModelExons ()
{
    if (currentTable.getSelectedItem().equals(modelExonLabel))
    {
        Gene_Model model;

        model = (Gene_Model)(Grail.geneModels.getCurrentItem());
        modelExonPanel.removeAll();
        if (model == null) title.setText("No Model Selected");
        else
        {
            title.setText("# of Exons: " + model.exons.num);
            modelExonPanel.add("Center",model.exons.textDisplay);
        }
        ((CardLayout)tablePanel.getLayout()).

```

```

        show(tablePanel,modelExonLabel);
        titlePanel.validate();
        modelExonPanel.validate();
    }
}

void reshow ()
{
    setTitle(mainTitle + ": Sequence: " + Grail.seqname +
        " Length: " + Grail.sequence.length());
    currentTable.select(grailExonLabel);
    setVersion();
    pack();
    show();
}

public boolean myHandleEvent (AWTEvent evt)
{
    try {
        int base;

        if (evt.getID() == AWTEvent.ITEM_STATE_CHANGED)
        {
            switch (evt.getStateChange())
            {
                case ItemEvent.SELECTED:

                    // Handle selection of a grail exon
                    if (currentTable.getSelectedItem().equals(grailExonLabel))
                    {
                        int itemSelected = Grail.exons[Grail.ver].itemList.
                            getSelectedIndex();
                        Exon exonSelected = (Exon) Grail.exons[Grail.ver].item.
                            elementAt(itemSelected);
                        base = ((exonSelected.dirn == 'f') ?
                            exonSelected.start-1 :
                            Grail.invertBaseNumber(exonSelected.end)-1);

                        // Update the main window
                        Grail.exons[Grail.ver].select();
                        Grail.mainCanvas.jumpToBase(base);

                        // Move the sequence window to the start of the new exon
                        Grail.sequenceWindow.sequenceCanvas.jumpToBase(base);
                        return true;
                    }
            }
        }
    }
}

```

```

// Handle selection of a model exon
else if (currentTable.getSelectedItem().
    equals(modelExonLabel))
{
    int currentModel = Grail.geneModels.itemList.
        getSelectedIndex();
    int itemSelected = ((Gene_Model)Grail.geneModels.item.
        elementAt(currentModel)).
        exons.itemList.getSelectedIndex();
    Model_Exon exonSelected = (Model_Exon)
        ((Gene_Model)Grail.
            geneModels.item.elementAt(currentModel)).exons.
            item.elementAt(itemSelected);

    base = (((Gene_Model)Grail.geneModels.item.elementAt(
        currentModel)).dirn == 'f') ?
        exonSelected.start-1 :
        Grail.invertBaseNumber(exonSelected.end)-1);

// Update the main window
((Gene_Model)Grail.geneModels.item.elementAt(
    currentModel)).exons.select();
Grail.mainCanvas.jumpToBase(base);

// Move the sequence window to the start of the new exon
Grail.sequenceWindow.sequenceCanvas.jumpToBase(base);

    return true;
}

// Handle selection of a new model
else if (currentTable.getSelectedItem().
    equals(geneModelLabel))
{
    Grail.mainCanvas.repaint();
    Grail.sequenceWindow.sequenceCanvas.repaint();
    return true;
}

else return false;

case ItemEvent.DESELECTED:
// On MacOS you sometimes get spurious LIST_DESELECT
// events. Only proceed here if no list item is
// selected.

```



```

        if (((List)evt.getSource()).getSelectedIndex() != -1)
            return true;

        // This event is for real so do our stuff
        if (currentTable.getSelectedItem().equals(grailExonLabel))
            Grail.exons[Grail.ver].deselectAll();
        else if (currentTable.getSelectedItem().
            equals(modelExonLabel))
            ((Gene_Model)Grail.geneModels.item.elementAt(
                Grail.geneModels.itemList.getSelectedIndex())).
                exons.deselectAll();
        Grail.mainCanvas.repaint();
        Grail.sequenceWindow.sequenceCanvas.repaint();
        return true;

        case WindowEvent.WINDOW_CLOSING:
            setVisible(false);
            return true;

        default: return false;
    }
}

        else
        {
            SHOW_GENE_MODELS: showModels(); return true;
        }

    }

// Pass any errors to the main grail window
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

return false;
}

public boolean myAction (ActionEvent evt)
{
    try {

        if (evt.getSource instanceof menuItem)
        {
            if (evt.getSource() == closeButton)
                setVisible(false);

```

```

return true;

    }
else if (evt.getSource() instanceof Choice)
{
    if (evt.getSource() == grailExonLabel) setVersion();
    else if (evt.getSource() == modelExonLabel) showModelExons();
    else if (evt.getSource() == geneModelLabel) showModels();
    return true;
}
}

// Pass any errors to the main grail window
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

return false;
}
}

```

```
////////////////////////////////////
```

```
// Cpg.java
```

```
//
```

```
// Next available error code: 31-3
```

```
////////////////////////////////////
```

```
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;
```

```
class Cpg extends GrailItem
```

```
{
```

```
    float score, GCcontent;
```

```
    static String buttonLabel, identifier, requestMessage, fieldNames;
```

```
    Cpg ()
```

```
{
```

```
    solid = false;
```

```
    color = Grail.cpgColor;
```

```
    highlightColor = Grail.cpgHighlightColor;
```

```
    dirn = 'f';
```

```
    buttonLabel = "CpG Island";
```

```
    identifier = "cpg";
```

```
    requestMessage = "dc.";
```

```
    fieldNames = " Start   End   Score %GC";
```

```
}
```

```
public String toString ()
```

```
{
```

```
    StringBuffer s = new StringBuffer(100);
```

```
    int i; for (i=0; i<50; i++) s.append(' ');
```

```
    try
```

```
{
```

```
        s.insert(0,start);
```

```
        s.insert(9,end);
```

```
        s.insert(18,score);
```

```
        s.insert(24,GCcontent);
```

```
}
```

```
    catch (ArrayIndexOutOfBoundsException e) {}
```

```

        catch (StringIndexOutOfBoundsException e) {}

        return s.toString().trim();
    }

    GraillItem newItem () { return new Cpg(); }

    String getLabel () { return buttonLabel; }

    String getIdentifier () { return identifier; }

    String getRequestMessage () { return requestMessage; }

    String getFieldNames () { return fieldNames; }

    void write (WriteStream out) throws ErrorException
    {
        out.writeBytes (int2str(start,6) + " " + int2str(end,6) + " " +
                        float2str(score,8,2) + " " + float2str(GCcontent,7,2) + "\n");
    }

    void read (ReadStream in) throws ErrorException
    {
        StringTokenizer st;

        // Read the first non-blank line
        st = new StringTokenizer(in.readFirstNonBlankLine());

        try {
            // Parse out the start position
            start = Integer.parseInt(st.nextToken());

            // Parse out the end position
            end = Integer.parseInt(st.nextToken());

            // Parse out the score
            score = Float.valueOf(st.nextToken()).floatValue();

            // Parse out the GC content
            GCcontent = Float.valueOf(st.nextToken()).floatValue();
        }

        catch (NumberFormatException e) {
            throw new ErrorException(e,"Error reading CpG island",31,1);
        }
    }

```

```
catch (NoSuchElementException e) {  
    throw new RuntimeException(e,"Error reading CpG island",31,2);  
}  
}  
}
```

```
////////////////////
// RecognitionException.java
////////////////////
```

```
import java.io.*;
import java.util.NoSuchElementException;
```

```
class RecognitionException extends Exception
```

```
{
    String message, code;
    Exception exception;
```

```
    RecognitionException (Exception e, String mesg, String ref)
```

```
    {
        super();
        message = mesg;
        code = ref;
        exception = e;
    }
```

```
    RecognitionException (Exception e)
```

```
    {
        this(e, "Description: \"" + e.toString() + "\"",
            "Message: \"" + e.getMessage() + "\"");
    }
```

```
    RecognitionException (Exception e, String mesg, int fn, int en, int ref)
```

```
    {
        this(e, mesg, "Ref: " + fn + "-" + en + "-" + ref);
    }
```

```
    RecognitionException (Exception e, String mesg, int fn, int en)
```

```
    {
        this(e, mesg, fn, en, 0);
    }
```

```
    RecognitionException (String mesg, String ref)
```

```
    {
        super();
        message = mesg;
        code = ref;
        exception = this;
    }
```

```

Exception (String mesg, int fn, int en, int ref)
{
    this(mesg,"Ref: " + fn + "-" + en + "-" + ref);
}

Exception (String mesg, int fn, int en)
{
    this(mesg,fn,en,0);
}

public String toString ()
{
    return message;
}

public String getMessage()
{
    return code;
}

String getStackTrace ()
{
    String trace = null;

    try {
        // Create a buffer to dump the stack trace into
        ByteArrayOutputStream buf = new ByteArrayOutputStream();

        // Create a printstream that uses the buffer
        PrintWriter stream = new PrintWriter(buf);

        // Dump the stack trace onto the print stream
        exception.printStackTrace(stream);

        // Flush the print stream
        stream.flush();

        // Flush the buffer
        buf.flush();

        // Get the stack trace
        trace = buf.toString();

        // Close the streams
        stream.close();
    }
}

```

```
        buf.close();
    }

    catch (IOException e) {
        trace = "Error getting stack trace";
    }

    // And return the trace
    return trace;
}
}
```



```

////////////////////
// ErrorWindow.java
////////////////////

```

```

import java.awt.*;
import java.awt.event.*;
import java.util.NoSuchElementException;

```

```

class ErrorWindow extends Frame
    implements ActionListener,
        KeyListener
{
    Panel flipper;
    boolean first;
    Button okButton;
    Thread callingThread;

    final String errorPanel = "1";
    final String stackPanel = "2";

    ErrorWindow (Exception e)
    {
        // Set the window title
        super("Error");

        // Make the panel that will display the error message
        Panel messagePanel = new Panel();
        messagePanel.setLayout(new GridLayout(4,1));
        messagePanel.add(new Label(""));
        Panel p = new Panel();
        p.add(new Label(e.toString()));
        messagePanel.add(p);
        p = new Panel();
        p.add(new Label(e.getMessage()));
        messagePanel.add(p);
        messagePanel.add(new Label(""));

        // Make the stack trace panel
        Panel tracePanel = new Panel();
        tracePanel.addKeyListener(this);
        tracePanel.setLayout(new BorderLayout());
        TextArea ta = new TextArea(e.getStackTrace());
        ta.setEditable(false);
    }
}

```

```

        tracePanel.add("Center",ta);

// Make the panel that will flip between error message and stack trace
flipper = new Panel();
flipper.setLayout(new CardLayout());
flipper.add(errorPanel,messagePanel);
flipper.add(stackPanel,tracePanel);

// Add the panels to the frame using default BorderLayout
add("Center",flipper);
p = new Panel();
p.add(okButton = new Button("OK"));
okButton.addActionListener(this);
add("South",p);

// Remember which panel is currently showing
((CardLayout)flipper.getLayout()).show(flipper,errorPanel);
first = true;

// Use pack() to size the window to best suit current machine
pack();
}

public void setVisible(boolean visvar)
{
    if (visvar)
    {
// Show myself
super.show();

// Find out what thread I'm in
callingThread = Thread.currentThread();

// The error window show() method blocks until the user hits the
// OK button. I originally implemented the blocking by calling
// suspend on the thread here, and calling resume on the thread
// in the action() method when they hit the OK button. However
// under netscape, calls to suspend() and resume() generate security
// exceptions. For this reason I just go to sleep here until
// the action() method sets the callingThread pointer to null
// (when the user hits the OK button.) I poll the callingThread
// pointer once every second and drop out of the loop as soon as
// I detect that it is null.
while (callingThread != null) {
    try { callingThread.sleep(1000); }
    catch (InterruptedException e) {}
}
}
}

```

```

    }
    }
    else
    {
        super.setVisible(false);
    }
}

public void actionPerformed (ActionEvent evt)
{
    if (evt.getSource() == okButton) {

        // They've hit OK so hide the window
        setVisible(false);

        // Tell the calling thread to resume. I originally called
        // suspend() on the calling thread and resumed it here. However
        // this generated a security exception under netscape so
        // I now just have the thread sleep until it detects that
        // the currentThread pointer has been set to null. Here I
        // set currentThread = null which will cause the show()
        // method to terminate and return.
        callingThread = null;

        // I've handled this event so return
    }
}

public void keyPressed (KeyEvent evt)
{
    // If they hit SHIFT-F1 toggle between the error message and
    // a non-editable text area that display the stack trace up
    // to the point where the exception was generated.

    if ( (evt.getKeyCode() == KeyEvent.VK_F1) && evt.isShiftDown()) {
        if (first)
            ((CardLayout)flipper.getLayout()).show(flipper,stackPanel);
        else
            ((CardLayout)flipper.getLayout()).show(flipper,errorPanel);
        first = !first;
    }
}

public void keyReleased (KeyEvent evt)
{

```

```
}
```

```
public void keyTyped (KeyEvent evt)
```

```
{
```

```
}
```

```
}
```

```
////////////////////////////////
```

```
// Exon.java
```

```
//
```

```
// Next available error code: 23-6
```

```
////////////////////////////////
```

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;
```

```
abstract class Exon extends GrailItem
```

```
{
```

```
    char qual;
```

```
    String shadow;
```

```
        float floatScore;
```

```
    int frame, orfStart, orfEnd, score;
```

```
    Translation trans;
```

```
    static String fieldNames;
```

```
    Exon ()
```

```
    {
```

```
        solid = true;
```

```
        highlightColor = Grail.exonHighlightColor;
```

```
        fieldNames = " Type Str Fr Position Op Rd Frm Scr Qual";
```

```
        trans = new Translation();
```

```
    }
```

```
    abstract int getVersion ();
```

```
    String getFieldNames () { return fieldNames; }
```

```
    boolean isShadow () { return shadow.equals("SHADOW: "); }
```

```
    public String toString ()
```

```
    {
```

```
        StringBuffer s = new StringBuffer(100);
```

```
        int i; for (i=0; i<46; i++) s.append(' ');
```

```

try
{
    s.insert(0,shadow);
    s.insert(11,dirn);
    s.insert(14,frame);
    s.insert(17,start);
    s.insert(26,end);
    s.insert(35,orfStart);
    s.insert(44,orfEnd);
    s.insert(53,score);
    s.insert(58,qual);
}
catch (ArrayIndexOutOfBoundsException e) {}
catch (StringIndexOutOfBoundsException e) {}

return s.toString().trim();
}

boolean mouseClicked (int base, char[] strand)
{
    if (dirn != strand[0]) return false;
    if (((strand[1] == 's') && isShadow()) ||
        ((strand[1] == 'a') && !isShadow())) return false;
    if (dirn == 'r') base = Grail.invertBaseNumber(base);
    return ((base >= start-1) && (base <= end+1));
}

void write (WriteStream out) throws Exception
{
    int i;

    if ((!Grail.fileCompMode) || (dirn == 'f')) {
        out.writeBytes (shadow + " " + dirn + " " + frame + " " +
            int2str(start,6) + " " + int2str(end,6) + " " +
                int2str(orfStart,6) + " " + int2str(orfEnd,6) + " " +
                    float2str(floatScore,7,2) + " " + qual + " " +
                        int2str(trans.start,6) + " " + int2str(trans.end,6) + "\n");
    }
    else {
        out.writeBytes (shadow + " " + dirn + " " + frame + " " +
            int2str(1+Grail.sequenceLength-end,6) + " " +
                int2str(1+Grail.sequenceLength-start,6) + " " +
                    int2str(1+Grail.sequenceLength-orfEnd,6) + " " +
                        int2str(1+Grail.sequenceLength-orfStart,6) + " " +
                            float2str(floatScore,7,2) + " " + qual + " " +

```

```

        int2str(1+Grail.sequenceLength-trans.end,6) + " " +
            int2str(1+Grail.sequenceLength-trans.start,6) + "\n");
    }
    out.writeBytes("[Translation = true]\n");
    out.writeBytes("[Start Translation]\n");
    for (i=0; i<trans.compressedText.length()/50; i++) {
        out.writeBytes(trans.compressedText.substring(
            i*50,(i+1)*50) + "\n");
    }
    if (trans.compressedText.length()%50 > 0) {
        out.writeBytes(trans.compressedText.substring(i*50) + "\n");
    }
    out.writeBytes("[End Translation]\n\n");
}

void read (ReadStream in) throws Exception
{
    String buf;
    StringTokenizer st;

    // Read the first non-blank line
    st = new StringTokenizer(in.readFirstNonBlankLine());

    try {
        // Get the type string
        shadow = st.nextToken();
        if (!shadow.equals("SHADOW:") && !shadow.equals("STANDARD:"))
            throw new Exception("Error reading exon",23,1);
        if (shadow.equals("SHADOW:")) shadow = "SHADOW: ";

        // grab the direction character
        if ((buf = st.nextToken()).length() != 1)
            throw new Exception("Error reading exon",23,2);
        dirn = buf.charAt(0);

        // parse out the frame
        frame = Integer.parseInt(st.nextToken());

        // parse out the start position
        start = Integer.parseInt(st.nextToken());

        // parse out the end position
        end = Integer.parseInt(st.nextToken());

        // parse out the ORF start position
        orfStart = Integer.parseInt(st.nextToken());
    }
}

```

```

// parse out the ORF end position
orfEnd = Integer.parseInt(st.nextToken());

// Get the exon score
floatScore = Float.valueOf(st.nextToken()).floatValue();
score = (int)floatScore;

// grab the quality character
if ((buf = st.nextToken()).length() != 1)
    throw new RuntimeException("Error reading exon",23,3);
qual = buf.charAt(0);

// get the translation start and end position
trans.start = Integer.parseInt(st.nextToken());
trans.end = Integer.parseInt(st.nextToken());

        //      Correct for complementary mode
        if ((Grail.compMode) && (dim == 'r')) {
            int tmp;
            tmp = 1 + Grail.sequenceLength - end;
            end = 1 + Grail.sequenceLength - start;
            start = tmp;
            tmp = 1 + Grail.sequenceLength - orfEnd;
            orfEnd = 1 + Grail.sequenceLength - orfStart;
            orfStart = tmp;
            tmp = 1 + Grail.sequenceLength - trans.end;
            trans.end = 1 + Grail.sequenceLength - trans.start;
            trans.start = tmp;
        }

        //      Read in the translation
        trans.read(in,dim);

// Set the exon color
color = (isShadow) ? Grail.shadowColor : Grail.exonColor;
}

catch (NumberFormatException e) {
    throw new RuntimeException(e,"Error reading exon",23,4);
}

catch (NoSuchElementException e) {
    throw new RuntimeException(e,"Error reading exon",23,5);
}
}

```



```

void draw (Rectangle b, int fsh, int rsh, int fth, int rth,
          float basesPerPixel, Graphics g, Rectangle clipRect)
// b.x = y position for reverse strand exon bars
// b.y = y position for forward strand exon bars
// b.width = X offset (i.e base number plotted at pixel x=0)
// b.height = height of bar
// fsh = forward strand shadow exon height
// rsh = reverse strand shadow exon height
// fth = forward strand translation height
// rth = reverse strand translation height
{
// Call GrailItem.draw() to draw the exon bar
if (isShadow()) { b.x = rsh; b.y = fsh; }
super.draw(b,basesPerPixel,g,clipRect);

// Draw the translation
if (selected) {
    g.setColor(color);
    trans.draw(b.width,fth,rth,dirn,basesPerPixel,g);
}
}

```

```

void draw (Rectangle b, Rectangle p, int fsh, int rsh, int ysf,
          float basesPerPixel, Graphics g, Rectangle clipRect)
// b.x = y position for reverse strand exon bars
// b.y = y position for forward strand exon bars
// b.width = X offset (i.e base number plotted at pixel x=0)
// b.height = height of bar
// p.x = y position for forward strand baseline
// p.y = y position for forward strand topline
// p.width = y position for reverse strand baseline
// p.height = y position for reverse strand topline
// fsh = forward strand shadow exon height
// rsh = reverse strand shadow exon height
// ysf = y scale factor
{
    int fy=b.y, ry=b.x;

// Call GrailItem.draw() to draw the exon bar
// This call also sets r.x and r.width to meaningful values
// (r is a GrailItem class variable)
if (isShadow()) { b.x = rsh; b.y = fsh; }
super.draw(b,basesPerPixel,g,clipRect);

// Bail now if this is a Grail 1 Exon

```

```

    if (getVersion() == 0) return;

    b.x = ry; b.y = fy;

    // Compute a bounding box for the plot lines
    if (dirn == 'f') r.y = p.y;
    else r.y = p.width;
    r.height = p.x+1-p.y;

    // Go ahead and plot the exon if it is visible
    // Shouldn't have to check for visibility but java can screw things
    // up sometimes if you don't
    if (r.intersects(clipRect))
    {
        // Set the color
        g.setColor(Color.white);

        // Set r.y to baseline
        r.y = ((dirn == 'f') ? p.x : p.width);

        // Set r.height to topline
        r.height = ((dirn == 'f') ?
            Math.max(r.y-score/ysf,p.y) :
            Math.min(r.y+score/ysf,p.height));

        // Draw the left line
        g.drawLine(r.x,r.y,r.x+1,r.height);

        // Draw the top line
        g.drawLine(r.x+1,r.height,r.x+r.width-2,r.height);

        // Draw the right line
        g.drawLine(r.x+r.width-2,r.height,r.x+r.width-1,r.y);

        // Blank out the baseline
        g.setColor(Color.black);
        g.drawLine(r.x+1,r.y,r.x+r.width-2,r.y);
    }
}
}

```

```

//////////
// Exon1.java
//////////

class Exon1 extends Exon
{
    static String buttonLabel, identifier, requestMessage;
    final static int version = 0;

    Exon1 ()
    {
        super();
        buttonLabel = "Grail 1 ";
        identifier = "Grail1_Exons";
        requestMessage = "d0T1.";
    }

    int getVersion () { return version; }

    GrailItem newItem () { return new Exon1(); }

    String getLabel () { return buttonLabel; }

    String getIdentifier () { return identifier; }

    String getRequestMessage () { return requestMessage; }
}

```

```

//////////
// Exon1a.java
//////////

class Exon1a extends Exon
{
    static String buttonLabel, identifier, requestMessage;
    final static int version = 1;

    Exon1a ()
    {
        super();
        buttonLabel = "Grail 1a";
        identifier = "Grail1a_Exons";
        requestMessage = "d1T.";
    }

    int getVersion () { return version; }

    GrailItem newItem () { return new Exon1a(); }

    String getLabel () { return buttonLabel; }

    String getIdentifier () { return identifier; }

    String getRequestMessage () { return requestMessage; }
}

```

```

//////////
// Exon2.java
//////////

class Exon2 extends Exon
{
    static String buttonLabel, identifier, requestMessage;
    final static int version = 2;

    Exon2 ()
    {
        super();
        buttonLabel = "Grail 2 ";
        identifier = "Grail2_Exons";
        requestMessage = "dgT.";
    }

    int getVersion () { return version; }

    GrailItem newItem () { return new Exon2(); }

    String getLabel () { return buttonLabel; }

    String getIdentifier () { return identifier; }

    String getRequestMessage () { return requestMessage; }
}

```

```
////////////////////////////////////
```

```
// ExonList.java
```

```
//
```

```
// Next available error code: 34-3
```

```
////////////////////////////////////
```

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.NoSuchElementException;
import java.util.StringTokenizer;
```

```
class ExonList extends GrailObject
```

```
{
```

```
    int version;
```

```
    boolean showingShadows;
```

```
    float[] forprob, revprob;
```

```
    int numprobs;
```

```
    ExonList (Exon sampleItem)
```

```
    {
```

```
        super(sampleItem);
```

```
        version = sampleItem.getVersion();
```

```
        showingShadows = true;
```

```
    }
```

```
    void showShadowExons ()
```

```
    {
```

```
        int i;
```

```
        // If they're already showing return immediately
```

```
        if (showingShadows) return;
```

```
        // Otherwise add the shadow exons to the list
```

```
        num = item.size();
```

```
        for (i=0; i<num; i++)
```

```
            if (((Exon)item.elementAt(i)).isShadow())
```

```
                itemList.addItem(((Exon)item.elementAt(i)).toString());
```

```
        // Update state to reflect new list
```

```

        showingShadows = true;
    }

    void hideShadowExons ()
    {
        int i,j;

        // If they're already hidden return immediately
        if (!showingShadows) return;

        // Otherwise delete the shadow exons from the list.
        // I originally used delItems() to remove the shadow
        // exons. However this does not work on MacOS. Instead
        // you have to clear the list totally and then reinsert
        // just the standard exons

        // Find out what item is currently selected, so we can
        // preserve it if it is a standard exon
        j = itemList.getSelectedIndex();

        // Wipe out the list
        itemList.removeAll();

        // Now re-insert the standard exons and deselect any shadow
        // exons so we don't get confused when we display them again
        for (i=0; i<num; i++)
            if (!((Exon)item.elementAt(i)).isShadow())
                itemList.addItem(((Exon)item.elementAt(i)).toString());
            else ((Exon)item.elementAt(i)).deselect();

        // Update state to reflect new list
        num = itemList.getItemCount();
        showingShadows = false;

        // Re-select whatever was previously selected (if appropriate)
        if ((j>=0) && (j<num)) itemList.select(j);

        // And validate to make sure the change shows up
        itemList.getParent().validate();
    }

    boolean pressButton () throws Exception
    {
        boolean retval;

        show();
    }

```

```

        if (!got()) retval = analyze();
        else retval = true;
        Grail.shadowExonsButton.setState(false);
        hideShadowExons();
        return retval;
    }

    void read (ReadStream in) throws Exception
    {
        if (version == 0) { /* READ GRAIL 1 PROBABILITIES */
            int i,j;
            String buf;
            StringTokenizer st;

            try {
                in.readMessage("[Start Grail1_forprob]");
                buf = in.readPrefix("[Num = ");
                numprobs = Integer.parseInt(buf.substring(0,buf.length()-1));
                forprob = new float[numprobs];
                revprob = new float[numprobs];
                for (i=0; i<numprobs; i+=10) {
                    st = new StringTokenizer(in.readFirstNonBlankLine());
                    for (j=0; (j<10)&&(i+j<numprobs); j++) {
                        forprob[i+j] = Float.valueOf(st.nextToken()).
                            floatValue();
                    }
                }
                in.readMessage("[End Grail1_forprob]");

                in.readMessage("[Start Grail1_revprob]");
                buf = in.readPrefix("[Num = ");
                numprobs = Integer.parseInt(buf.substring(0,buf.length()-1));
                for (i=0; i<numprobs; i+=10) {
                    st = new StringTokenizer(in.readFirstNonBlankLine());
                    for (j=0; (j<10)&&(i+j<numprobs); j++) {
                        revprob[i+j] = Float.valueOf(st.nextToken()).
                            floatValue();
                    }
                }
                in.readMessage("[End Grail1_revprob]");
            }

            catch (NumberFormatException e) {
                throw new Exception(e,"Error reading exons",34,1);
            }
        }
    }

```



```

        catch (NoSuchElementException e) {
            throw new ErrorException(e, "Error reading exons", 34, 2);
        }
    }

    super.read(in);
    showingShadows = true;
    hideShadowExons();
}

void write (WriteStream out) throws ErrorException
{
    if (version == 0) {
        int i, j;
        String prob;

        out.writeBytes("[Start Grail1_forprob]\n");
        out.writeBytes("[Num = " + numprobs + "]\n");
        for (i=0; i<numprobs; i+=10) {
            for (j=0; (j<10)&&(i+j<numprobs); j++) {
                prob = Float.toString(forprob[i+j]);
                switch (prob.length()) {
                    case 1: prob = prob + ".000"; break;
                    case 3: prob = prob + "00"; break;
                    case 4: prob = prob + "0"; break;
                    default: break;
                }
                out.writeBytes(prob + " ");
            }
            out.writeBytes("\n");
        }
        out.writeBytes("[End Grail1_forprob]\n\n");

        out.writeBytes("[Start Grail1_revprob]\n");
        out.writeBytes("[Num = " + numprobs + "]\n");
        for (i=0; i<numprobs; i+=10) {
            for (j=0; (j<10)&&(i+j<numprobs); j++) {
                prob = Float.toString(revprob[i+j]);
                switch (prob.length()) {
                    case 1: prob = prob + ".000"; break;
                    case 3: prob = prob + "00"; break;
                    case 4: prob = prob + "0"; break;
                    default: break;
                }
                out.writeBytes(prob + " ");
            }
        }
    }
}

```

```

        out.writeBytes("\n");
    }
    out.writeBytes("[End Grail1_revprob]\n\n");
}

super.write(out);
}

void draw (Rectangle b, int fsh, int rsh, int fth, int rth,
          float basesPerPixel, Graphics g, Rectangle clipRect)
// b.x = y position for reverse strand exon bars
// b.y = y position for forward strand exon bars
// b.width = X offset (i.e base number plotted at pixel x=0)
// b.height = height of bar
// fsh = forward strand shadow exon height
// rsh = reverse strand shadow exon height
// fth = forward strand translation height
// rth = reverse strand translation height
{
    int i;

    for (i=0; i<num; i++)
    {
        ((Exon)item.elementAt(i)).
            draw(b,fsh,rsh,fth,rth,basesPerPixel,g,clipRect);
    }
}

void draw (Rectangle b, Rectangle p, int fsh, int rsh, int ysf,
          float basesPerPixel, Graphics g, Rectangle clipRect)
// b.x = y position for reverse strand exon bars
// b.y = y position for forward strand exon bars
// b.width = X offset (i.e base number plotted at pixel x=0)
// b.height = height of bar
// p.x = y position for forward strand baseline
// p.y = y position for forward strand topline
// p.width = y position for reverse strand baseline
// p.height = y position for reverse strand topline
// fsh = forward strand shadow exon height
// rsh = reverse strand shadow exon height
// ysf = y scale factor
{
    int x2, i,k;

    // Draw the two base lines
    x2 = (int)((Grail.sequenceLength-b.width)/basesPerPixel);

```

```

g.setColor(Color.white);
g.drawLine(0,p.x,x2,p.x);
g.drawLine(0,p.width,x2,p.width);

// Handle Grail 1 Exons
if (version == 0) {
    double dx1,dx2,width=10/basesPerPixel;
    int P,y1,y2,maxx,height=p.y-p.x;

    if ((P=(b.width-45)/10) < 0) P = 0;
    dx1 = (45+P*10)/basesPerPixel;
    y1 = (int)(p.x+forprob[P]*height);
    maxx = clipRect.x + clipRect.width;
    if ((dx2 = (numprobs*10+45)/basesPerPixel) > maxx) dx2 = maxx;
    g.setColor(Color.black);
    g.drawLine((int)dx1,p.x,(int)dx2,p.x);
    g.drawLine((int)dx1,p.width,(int)dx2,p.width);
    g.setColor(Color.white);
    dx2 = dx1+width;
    for (; (P<numprobs-1) && (dx1<maxx); P++,dx1=dx2,dx2+=width) {
        y2 = (int)(p.x + forprob[P+1]*height);
        g.drawLine((int)dx1,y1,(int)dx2,y2);
        y1 = y2;
    }

    if ((P=(b.width-45)/10) < 0) P = 0;
    height = p.height-p.width;
    dx1 = (45+P*10)/basesPerPixel;
    y1 = (int)(p.width+revprob[P]*height);
    dx2 = dx1+width;
    for (; (P<numprobs-1) && (dx1<maxx); P++,dx1=dx2,dx2+=width) {
        y2 = (int)(p.width + revprob[P+1]*height);
        g.drawLine((int)dx1,y1,(int)dx2,y2);
        y1 = y2;
    }
}

// Draw the exons
for (i=0; i<num; i++)
{
    ((Exon)item.elementAt(i)).
        draw(b,p,fsh,rsh,ysf,basesPerPixel,g,clipRect);
}
}

```

```

//////////
// FeaturesWindow.java
//////////

import java.awt.*;
import java.awt.event.*;
import java.awt.AWTEvent;

class FeaturesWindow extends ThreadedFrame
{
    final static String title = "FEATURES";
    final static String buttonName = "Features";

    MenuItem printButton, closeButton;
    Panel buttonPanel, tablePanel;
    CheckboxGroup cbg;
    Grail myParent;

    FeaturesWindow (Grail g)
    {
        super(title);
        setLayout(new BorderLayout());

        // Store pointer to Grail
        myParent = g;

        // Set up the menus
        printButton = new MenuItem("Print"); printButton.setEnabled(false);
        closeButton = new MenuItem("Close");
        closeButton.addActionListener(this);
        Menu fileMenu = new Menu("File");
        fileMenu.add(printButton);
        fileMenu.addSeparator();
        fileMenu.add(closeButton);
        MenuBar mb = new MenuBar();
        mb.add(fileMenu);
        setMenuBar(mb);

        // Create the Radio Buttons panel
        buttonPanel = new Panel();
        buttonPanel.addItemListener(this);
        buttonPanel.setLayout(new GridLayout(4,1));
        cbg = new CheckboxGroup();
        cbg.addItemListener(this);
        buttonPanel.add(new Checkbox(Grail.polya.button.getLabel(),cbg,true));
        Grail.polya.button.addItemListener(grail);
    }
}

```

```

        buttonPanel.add(new Checkbox(Grail.promoter.button.getLabel(),cbg,false));
        Grail.promoter.button.addItemListener(grail);
        buttonPanel.add(new Checkbox(Grail.cpg.button.getLabel(),cbg,false));
        Grail.cpg.button.addItemListener(grail);
        buttonPanel.add(new Checkbox(Grail.rpt.button.getLabel(),cbg,false));
        Grail.rpt.button.addItemListener(grail);
        add("West",buttonPanel);

// Create the table panel
tablePanel = new Panel();
tablePanel.setLayout(new CardLayout());
tablePanel.add(Grail.polya.button.getLabel(),
    Grail.polya.textDisplay);
tablePanel.add(Grail.promoter.button.getLabel(),
    Grail.promoter.textDisplay);
tablePanel.add(Grail.cpg.button.getLabel(),
    Grail.cpg.textDisplay);
tablePanel.add(Grail.rpt.button.getLabel(),
    Grail.rpt.textDisplay);
((CardLayout)tablePanel.getLayout()).
    show(tablePanel,Grail.polya.button.getLabel());
add("Center",tablePanel);

// Call pack() to size the thing
pack();
}

boolean selectItem (GrailObject g, String label)
{
    if (label.equals(g.button.getLabel()))
    {
        g.select();
        if (g.isShowing()) {
            Grail.mainCanvas.jumpToBase(
                g.getCurrentItem().forwardStartBase());
            Grail.sequenceWindow.sequenceCanvas.jumpToBase(
                g.getCurrentItem().forwardStartBase());
        }
        return true;
    }
    else return false;
}

boolean deselectItem (GrailObject g, String label)
{

```

```

    if (label.equals(g.button.getLabel()))
    {
        g.deselectAll();
        if (g.isShowing()) {
            Grail.mainCanvas.repaint();
            Grail.sequenceWindow.sequenceCanvas.repaint();
        }
        return true;
    }

    else return false;
}

void reshow ()
{
    setTitle(title + ": Sequence: " + Grail.seqname +
        " Length: " + Grail.sequence.length());
    pack();
    show();
}

public boolean myHandleEvent (AWTEvent evt)
{
    try {
        String label = cbg.getSelectedCheckbox().getLabel();

        switch (evt.getStateChange())
        {
            case ItemEvent.SELECTED:

                if (selectItem(Grail.promoter,label));
                else if (selectItem(Grail.cpg,label));
                else if (selectItem(Grail.polya,label));
                else selectItem(Grail.rpt,label);
                return true;

            case ItemEvent.DESELECTED:
                // On MacOS you sometimes get spurious LIST_DESELECT
                // events. Only proceed here if no list item is
                // selected.
                if (((List)evt.getSource()).getSelectedIndex() != -1)
                    return true;

                // This event is for real so do our stuff
                if (deselectItem(Grail.promoter,label));
                else if (deselectItem(Grail.cpg,label));

```

```

        else if (deselectItem(Grail.polya,label));
        else deselectItem(Grail.rpt,label);
        return true;

    case WindowEvent.WINDOW_CLOSING:
        setVisible(false);
        return true;

    default: return false;
}
}

// Pass any errors to grail main window
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

return false;
}

public boolean myAction (AWTEvent evt)
{
    try {
        if (evt.getSource() instanceof Checkbox)
        {
            ((CardLayout)tablePanel.getLayout()).
                show(tablePanel,((Checkbox)evt.getSource()).getLabel());
            return true;
        }

        else if (evt.getSource() == closeButton)
        {
            setVisible(false);
            return true;
        }
    }

    // Pass any errors to grail main window
    catch (Exception e) {
        myParent.dispatchEvent(new ExceptionEvent(myParent,e));
    }

    return false;
}
}

```

```
////////////////////////////////////
```

```
// FileChoice.java
```

```
//
```

```
// Next available error code: 19-5
```

```
////////////////////////////////////
```

```
import java.awt.*;  
import java.util.Vector;  
import java.awt.event.*;
```

```
class FileChoice extends ThreadedFrame
```

```
{
```

```
    final static int CANCEL = 0;
```

```
    final static int LOAD = 1;
```

```
    final static int SAVE = 2;
```

```
    int mode;
```

```
    AppletServer fileServer;
```

```
    String wd, mask, filter, selection;
```

```
    TextField filterBox, selectionBox;
```

```
    List dirList, fileList;
```

```
    Button openButton, cancelButton, filterButton;
```

```
    Thread callingThread;
```

```
FileChoice (int type, AppletServer server) throws ErrorException
```

```
{
```

```
    super(type == LOAD ? "Load" : "Save");
```

```
    fileServer = server;
```

```
    mode = type;
```

```
// Make the panel that holds the filter box
```

```
    Panel filterPanel = new Panel();
```

```
// Replace the panel's default FlowLayout with a BorderLayout
```

```
    filterPanel.setLayout(new BorderLayout());
```

```
// Add the "Filter" label which, by default, aligns to the left
```

```
    filterPanel.add("North", new Label("Filter"));
```

```
// Add the filter TextField with a minimum of 16 columns
```

```
    filterPanel.add("South", filterBox = new TextField(16));
```

```
    filterBox.setEditable(false);
```

```
// Make the panel that holds the directories list
```

```
    Panel dirPanel = new Panel();
```



```

// Replace the panel's default FlowLayout with a BorderLayout
dirPanel.setLayout(new BorderLayout());
// Add the "Directories" label which, by default, aligns to the left
dirPanel.add("North",new Label("Directories"));
// Add the directories list with 10 items visible
// and multiple selections not allowed
dirPanel.add("Center",dirList = new List(10,false));

dirPanel.addItemListener(this);

// Make the panel that holds the file list
Panel filePanel = new Panel();
// Replace the panel's default FlowLayout with a BorderLayout
filePanel.setLayout(new BorderLayout());
// Add the "Files" label which, by default, aligns to the left
filePanel.add("North",new Label("Files"));
// Add the files list with 10 items visible
// and multiple selections not allowed
filePanel.add("Center",fileList = new List(10,false));

filePanel.addItemListener(this);

// Make the panel that holds the listings
Panel listPanel = new Panel();
// Replace the panel's default FlowLayout with a BorderLayout
listPanel.setLayout(new BorderLayout());
// Add the directory panel to the center so it takes up
// any extra space
listPanel.add("Center",dirPanel);
// Add the file panel at the right edge
listPanel.add("East",filePanel);

// Make the panel that holds the selection box
Panel selectionPanel = new Panel();
// Replace the panel's default FlowLayout with a BorderLayout
selectionPanel.setLayout(new BorderLayout());
// Add the "Selection" label which, by default, aligns to the left
selectionPanel.add("North",new Label("Selection"));
// Add the selection TextField with a minimum of 16 columns
selectionPanel.add("South",selectionBox = new TextField(16));
selectionBox.setEditable(true);
selectionPanel.addActionListener(this);

selectionPanel.addItemListener(this);

```

```

// Make the panel that holds the buttons
    Panel buttonPanel = new Panel();
// Replace default FlowLayout with a 1-row, 3-column grid
    buttonPanel.setLayout(new GridLayout(1,3));
// Place "Open" button in its own panel and add to the button panel
    Panel tempPanel = new Panel();
    tempPanel.add(openButton = new Button("Open"));
    buttonPanel.add(tempPanel);
    buttonPanel.addActionListener(this);
    openButton.addActionListener(this);

// Place "Filter" button in its own panel and add to the button panel
    tempPanel = new Panel();
    tempPanel.add(filterButton = new Button("Filter"));
    buttonPanel.add(tempPanel);
// Place "Cancel" button in its own panel and add to the button panel
    tempPanel = new Panel();
    tempPanel.add(cancelButton = new Button("Cancel"));
    buttonPanel.add(tempPanel);
    tempPanel.addActionListener(this);
    cancelButton.addActionListener(this);
    filterButton.addActionListener(this);

// Make the lower panel
    Panel lowerPanel = new Panel();
// Replace the panel's default FlowLayout with a BorderLayout
    lowerPanel.setLayout(new BorderLayout());
// Add the selection panel and button panel
    lowerPanel.add("North",selectionPanel);
    lowerPanel.add("South",buttonPanel);

// Put it all together using the Frame's default BorderLayout
    add("North",filterPanel);
    add("Center",listPanel);
    add("South",lowerPanel);

// Set the file mask
    mask = "*";

// Retrieve the current listing from the server
    getListing();

// Finally, use pack to resize the window
    pack();
}

```

```

String filterFile (String file)
{
    return file;
}

void getListing () throws Exception
{
    String dirent;
    Vector files;
    int i,n;

    // Retrieve the current listing from the server
    files = fileServer.ls();

    // Make sure we have a non-empty listing
    if ((n = files.size()) <= 0)
        throw new Exception("Error retrieving listing",19,1);

    // Get the current directory - it will always end with the
    // appropriate system-dependant file separator character
    wd = ((String)files.elementAt(0));

    // Clear the directories and files lists
    if (dirList.getItemCount() > 0) dirList.removeAll();
    if (fileList.getItemCount() > 0) fileList.removeAll();

    // Build the directories and files lists
    dirList.addItem(".");
    dirList.addItem("..");
    for (i=1; i<n; i++) {
        // Get the next directory entry
        dirent = (String)files.elementAt(i);

        // If its a directory, add it to dirList. Directories are
        // returned with a '/' character appended. This is removed
        // before the directory is displayed in the dirList.
        if (dirent.endsWith("/"))
            dirList.addItem(dirent.substring(0,dirent.length()-1));

        // If its a file filter it before adding to fileList
        else if ((dirent = filterFile(dirent)) != null)
            fileList.addItem(dirent);
    }

    // Make sure no items are selected
    if ((i = dirList.getSelectedIndex()) != -1) dirList.deselect(i);
}

```

```

        if ((i = fileList.getSelectedIndex()) != -1) fileList.deselect(i);

// Set the current selection and filter
    selectionBox.setText(selection = wd);
    filterBox.setText(wd + mask);

// Use pack to adjust the window size to best suit the local system
    validate(); pack();
}

public boolean actionNeedsA Thread (ActionEvent evt)
{
    return ((evt.getSource() instanceof Button)||
            (evt.getSource() instanceof List) ||
            (evt.getSource() instanceof TextField));
}

public boolean myHandleEvent (AWTEvent evt)
{
// Handle directory selections
    if (evt.getSource() == dirList) {
        // This prevents the user from deselecting
        if (evt.getStateChange() == ItemEvent.DESELECTED) {
            dirList.select(((Integer)evt.getItem()).intValue());
            return true;
        }
    }

// Handle file selections
    else if (evt.getSource() == fileList) {
        if (evt.getStateChange() == ItemEvent.SELECTED) {
            selection = wd + fileList.
                getItem(((Integer)evt.getItem()).intValue());
            selectionBox.setText(selection);
            return true;
        }

// This prevents the user from deselecting
        else if (evt.getStateChange() == ItemEvent.DESELECTED) {
            fileList.select(((Integer)evt.getItem()).intValue());
            return true;
        }
    }
}

```

```

        return false;
    }

    public void threadedAction (AWTEvent evt) throws Exception
    {
        // Deal with the cancel button
        if (evt.getSource() == cancelButton) {

            // Hide the window - the main frame will dispose of it
            hide(CANCEL);
        }

        // Deal with the open button
        else if (evt.getSource() == openButton) {

            // If there is no file selected then barf
            if (fileList.getSelectedIndex() == -1)
                throw new Exception (
                    "No file selected for opening",19,2);

            // If they did select a file then hide window and return
            else {
                hide(mode);
            }
        }

        // Deal with the Selection TextField
        else if (evt.getSource() == selectionBox) {
            // If there is no file selected then barf
            if (fileList.getSelectedIndex() == -1)
                throw new Exception(
                    "No file selected for opening",19,3);

            // If they did select a file then hide window and return
            else {
                hide(mode);
            }
        }

        // Deal with the filter button
        else if (evt.getSource() == filterButton) {

            if (dirList.getSelectedIndex() != -1) {

                // Change to the new directory
            }
        }
    }

```

```

        fileServer.cd(dirList.getSelectedItem());

        // And get the listing
        getListing();
    }
}

// Deal with a DOUBLE CLICK on the Directory list
else if (evt.getSource() == dirList) {

    // Change to the new directory
    fileServer.cd(dirList.getSelectedItem());

    // And get the listing
    getListing();
}

// Deal with a DOUBLE CLICK on the File list
else if (evt.getSource() == fileList) {

    // First, do the same thing as for a SELECT event
    selection = wd + (String)((ItemEvent)evt).getItem().toString();
    selectionBox.setText(selection);

    // Then hide and return, since a file has been selected
    hide(mode);
}
}

public void setVisible(boolean visvar)
{
    if (visvar)
    {
        // Show the window
        super.setVisible(true);

        // Find out what thread I'm in
        callingThread = Thread.currentThread();

        // The fileChoice show() method blocks until the user selects a file
        // or cancels. I originally implemented the blocking by calling
        // suspend on the thread here, and calling resume on the thread
        // in the action() method when they hit the OK button. However
        // under netscape, calls to suspend() and resume() generate security
        // exceptions. For this reason I just go to sleep here until

```

```
// the action() method sets the callingThread pointer to null
// (when the user hits the OK button.) I poll the callingThread
// pointer once every second and drop out of the loop as soon as
// I detect that it is null.
```

```
    while (callingThread != null) {
        try { callingThread.sleep(1000); }
        catch (InterruptedException e) {}
    }
    else
    {
        super.setVisible(false);
    }
}
```

```
public void hide (int md)
{
    mode = md;
    setVisible(false);
```

```
// Tell the calling thread to resume. I originally called
// suspend() on the calling thread and resumed it here. However
// this generated a security exception under netscape so
// I now just have the thread sleep until it detects that
// the currentThread pointer has been set to null. Here I
// set currentThread = null which will cause the show()
// method to terminate and return.
    callingThread = null;
}
```

```
String getSelection ()
{
    return ((mode == CANCEL) ? null : selectionBox.getText());
}
}
```

```
////////////////////////////////////
```

```
// Gene_Model.java
```

```
//
```

```
// Next available error code: 24-4
```

```
////////////////////////////////////
```

```
import java.awt.Graphics;  
import java.awt.Rectangle;  
import java.io.IOException;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.util.StringTokenizer;  
import java.util.NoSuchElementException;
```

```
class Gene_Model extends GrailItem
```

```
{
```

```
    int score;
```

```
    GrailObject exons;
```

```
    static String buttonLabel, identifier, requestMessage, fieldNames;
```

```
    Gene_Model ()
```

```
{
```

```
        color = Grail.modelColor;
```

```
        buttonLabel = "Assemble";
```

```
        identifier = "Gene_Models";
```

```
        requestMessage = "daa.";
```

```
        fieldNames = "Str  Seq Range  Score Exons";
```

```
        exons = new GrailObject(new Model_Exon());
```

```
        exons.itemList.setEnabled(true);
```

```
}
```

```
    boolean mouseClicked (int base, char[] strand)
```

```
{
```

```
        return exons.mouseClicked(base,strand);
```

```
}
```

```
    void draw (Rectangle t, int fth, int rth, float basesPerPixel,  
               Graphics g, Rectangle clipRect)
```

```
    // t.x = y position for reverse strand plots
```

```
    // t.y = y position for forward strand plots
```

```
    // t.width = X offset (i.e base number plotted at pixel x=0)
```

```
    // t.height = height of bar
```



```

// fth = forward strand translation height
// rth = reverse strand translation height
{
    if (exons.item.size() == 0) return;

    int i;
    Model_Exon g2;

    for (i=0; i<exons.item.size(); i++) {
        g2 = (Model_Exon)(exons.item.elementAt(i));
        g2.draw(t,fth,rth,basesPerPixel,g,clipRect);
    }
}

void draw (Rectangle t, float basesPerPixel,
    Graphics g, Rectangle clipRect)
// t.x = y position for reverse strand plots
// t.y = y position for forward strand plots
// t.width = X offset (i.e base number plotted at pixel x=0)
// t.height = height of bar
{
    int i,s,e,m;
    GraillItem g1,g2;

    if (exons.item.size() == 0) return;

    if (dirn == 'f')
    {
        g1 = (Model_Exon)(exons.item.elementAt(0));
        g1.draw(t,basesPerPixel,g,clipRect);
        for (i=1; i<exons.item.size(); i++)
        {
            g2 = (Model_Exon)(exons.item.elementAt(i));
            g2.draw(t,basesPerPixel,g,clipRect);
            s = (int)((g1.end-t.width)/basesPerPixel);
            e = (int)((g2.start-t.width)/basesPerPixel);
            m = (s+e)/2;
            r.x = s; r.y = t.y; r.width = e-s; r.height = t.height;
            g.setColor(color);
            g.drawLine(r.x,r.y,m,r.y-r.height);
            g.drawLine(m,r.y-r.height,e,r.y);
            g1 = g2;
        }
    }
    else
    {

```

```

g1 = (Model_Exon)(exons.item.lastElement());
g1.draw(t,basesPerPixel,g,clipRect);
for (i=exons.item.size()-2; i>=0; i--)
{
    g2 = (Model_Exon)(exons.item.elementAt(i));
    g2.draw(t,basesPerPixel,g,clipRect);
    s = (int)((Grail.invertBaseNumber(g1.start)-t.width)/
        basesPerPixel);
    e = (int)((Grail.invertBaseNumber(g2.end)-t.width)/
        basesPerPixel);
    m = (s+e)/2;
    r.x = e; r.y = t.x; r.width = s-e; r.height = t.height;
    g.setColor(color);
    g.drawLine(r.x,r.y+r.height,m,r.y+2*r.height);
    g.drawLine(m,r.y+2*r.height,s,r.y+r.height);
    g1 = g2;
}
}
}

```

```

GrailItem newItem () { return new Gene_Model(); }

```

```

String getLabel () { return buttonLabel; }

```

```

String getIdentifier () { return identifier; }

```

```

String getRequestMessage () { return requestMessage; }

```

```

String getFieldNames () { return fieldNames; }

```

```

public String toString ()
{
    StringBuffer s = new StringBuffer(60);
    int i; for (i=0; i<30; i++) s.append(' ');

    try
    {
        s.insert(1,dirn);
        s.insert(5,start);
        s.insert(14,end);
        s.insert(23,score);
        s.insert(30,exons.num);
    }
    catch (ArrayIndexOutOfBoundsException e) {}
    catch (StringIndexOutOfBoundsException e) {}
}

```

```

        return s.toString().trim();
    }

    void write (WriteStream out) throws Exception
    {
        if ((!Grail.fileCompMode) || (dirn == 'f')) {
            out.writeBytes (dirn + " " + int2str(start,6) + " " +
                           int2str(end,6) + " " + int2str(score,5) + "\n");
        }
        else {
            out.writeBytes (dirn + " " +
                           int2str(1+Grail.sequenceLength-end,6) + " " +
                           int2str(1+Grail.sequenceLength-start,6) + " " +
                           int2str(score,5) + "\n");
        }
        exons.write(out);
    }

    void read (ReadStream in) throws Exception
    {
        int i;
        String buf;
        StringTokenizer st;
        Model_Exon m;

        // Read the first non-blank line
        st = new StringTokenizer(in.readFirstNonBlankLine());

        try {
            // grab the direction character
            if ((buf = st.nextToken()).length() != 1)
                throw new Exception("Error reading gene model",24,1);
            Grail.modelDirn = dirn = buf.charAt(0);

            // Parse out the start position
            start = Integer.parseInt(st.nextToken());

            // Parse out the end position
            end = Integer.parseInt(st.nextToken());

            // Grab the score
            score = Integer.parseInt(st.nextToken());

            // Correct for complementary mode
            if ((Grail.compMode) && (dirn == 'r')) {
                int tmp;

```

```

        tmp = 1 + Grail.sequenceLength - end;
        end = 1 + Grail.sequenceLength - start;
        start = tmp;
    }

    // Read in the exons
    exons.read(in);

    // Do the exon translations and set their direction character
    for (i=exons.item.size()-1; i>=0; i--)
    {
        m = (Model_Exon)(exons.item.elementAt(i));
        while (((m.trans.start-1)%3) != m.frame) m.trans.start++;
        if (dirn == 'r')
            while ((m.trans.end%3) != m.frame) m.trans.end--;
        m.trans.doTranslation(dirn);
    }
}

catch (NumberFormatException e) {
    throw new RuntimeException(e,"Error reading gene model",24,2);
}

catch (NoSuchElementException e) {
    throw new RuntimeException(e,"Error reading gene model",24,3);
}
}
}

```

```

//////////
// GeneModelList.java
//////////

import java.awt.Event;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.*;
import java.awt.AWTEvent;

class GeneModelList extends GrailObject
{
    GeneModelList (Gene_Model sampleItem)
    {
        super(sampleItem,true);
    }

    void draw (Rectangle r, int fth, int rth, float basesPerPixel,
        Graphics g, Rectangle clipRect)
    // r.x = y position for reverse strand plots
    // r.y = y position for forward strand plots
    // r.width = X offset (i.e base number plotted at pixel x=0)
    // r.height = height of bar
    // fth = forward strand translation height
    // rth = reverse strand translation height
    {
        Gene_Model gm = (Gene_Model)getCurrentItem();

        if (gm != null) gm.draw(r,fth,rth,basesPerPixel,g,clipRect);
    }

    void draw (Rectangle r, float basesPerPixel,
        Graphics g, Rectangle clipRect)
    // r.x = y position for reverse strand plots
    // r.y = y position for forward strand plots
    // r.width = X offset (i.e base number plotted at pixel x=0)
    // r.height = height of bar
    {
        GrailItem gm = getCurrentItem();

        if (gm != null) gm.draw(r,basesPerPixel,g,clipRect);
    }

    boolean mouseClicked (int base, char[] strand)

```

```

{
    GrailItem gm = getCurrentItem();

    if (gm == null) return false;

    return gm.mouseClick(base,strand);
}

boolean pressButton (int lo, int hi) throws Exception
{
    boolean retval;

    showing = true;
    requestMessage = itemClass.getRequestMessage() +
        Integer.toString(lo) + "\n" + Integer.toString(hi) + "\n";

    if (retval = analyze()) {

        Grail.analysisWindow.dispatchEvent(AWTEvent(Grail.analysisWindow,
            AnalysisWindow.SHOW_GENE_MODELS));

    }
    Grail.sequenceWindow.sequenceCanvas.repaint();
    Grail.mainCanvas.repaint();

    return retval;
}
}

```

```
////////////////////////////////////
```

```
// Grail.java
```

```
//
```

```
// Next available error code: 20-11
```

```
////////////////////////////////////
```

```
import java.io.*;
```

```
import java.awt.*;
```

```
import java.applet.Applet;
```

```
import java.awt.event.*;
```

```
class Grail extends ThreadedFrame
```

```
{
```

```
    static final int ERROR = -1;
```

```
    static boolean amApplet;
```

```
    ACTG applet;
```

```
    static String remoteHost, forwarderHost, fileHost;
```

```
    static int remotePort, forwarderPort, localPort;
```

```
    final int MAX_STRLEN = 81;
```

```
    final int MAX_SEQLEN = 100000;
```

```
    final int DEFAULT_VERSION = 2;
```

```
    final static String fileHeader = "[Start Grail]";
```

```
    final static int GC_SHIFT_SIZE = 10;
```

```
    final static int GC_WINDOW_SIZE = 50;
```

```
    final static String title = "ACTG";
```

```
    final static Color exonColor = Color.cyan;
```

```
    final static Color shadowColor = Color.lightGray;
```

```
    final static Color exonHighlightColor = Color.yellow;
```

```
    final static Color modelColor = Color.yellow;
```

```
    final static Color modelHighlightColor = Color.cyan;
```

```
    final static Color promoterColor = Color.pink;
```

```
    final static Color promoterHighlightColor = Color.cyan;
```

```
    final static Color cpgColor = Color.magenta;
```

```
    final static Color cpgHighlightColor = Color.yellow;
```

```
    final static Color polyaColor = Color.green;
```

```
    final static Color polyaHighlightColor = Color.yellow;
```

```

final static Color rptColor = Color.red;
final static Color rptHighlightColor = Color.cyan;

static Menu grailMenu, assembleMenu;
static Menu fileMenu, featuresMenu, windowsMenu;
static CheckboxMenuItem shadowExonsButton, gcButton;
static MenuItem loadButton, saveButton, exitButton;
static MenuItem seqWinButton, analWinButton, featWinButton;
static String loadFilename = null, saveFilename;

static MainCanvas mainCanvas;

static int ver;

static String seqname;
static String sequence, rsequence;
static int sequenceLength;
static Color[] GC;
static ExonList[] exons;
static GrailObject polyA, cpG, promoter, rpt;
static GeneModelList geneModels;

static AnalysisWindow analysisWindow;
static SequenceWindow sequenceWindow;
static FeaturesWindow featuresWindow;

boolean licenseExpired = false;
    static boolean fileCompMode = false;
    static boolean compMode = false;
    static char modelDirn;

Grail (ACTG parent, String mainServerHostname, int mainServerPort,
      String fileServerHostname, int userID, int fwdPort)
{
    super(title);

    // store the execution environment (if the userID and fwdPort
    // are both zero then we're an application.)
    amApplet = ((userID != 0) || (fwdPort != 0));
    applet = parent;

    // Initialize the hosts and ports through which we will
    // communicate with the servers
    remoteHost = mainServerHostname;
    remotePort = mainServerPort;
    fileHost = fileServerHostname.toLowerCase();

```



```

// If we're an applet get forwarder and local server info
    if (amApplet) {
        localPort = userID;
        forwarderPort = fwdPort;
        forwarderHost = applet.getCodeBase().getHost();
    }

    MenuBar mb = new MenuBar();

// Initialize some variables
    exons = new ExonList[3];
    exons[0] = new ExonList(new Exon1());
    exons[1] = new ExonList(new Exon1a());
    exons[2] = new ExonList(new Exon2());
    polya = new GrailObject(new PolyA());
    cpg = new GrailObject(new Cpg());
    promoter = new GrailObject(new Prom());
    rpt = new GrailObject(new Rpt());
    geneModels = new GeneModelList(new Gene_Model());

// Disable the feature lists
    polya.itemList.setEnabled(false);
    promoter.itemList.setEnabled(false);
    rpt.itemList.setEnabled(false);
    cpg.itemList.setEnabled(false);

// Construct the file menu
    fileMenu = new Menu("File");
    fileMenu.add(loadButton = new MenuItem("Load"));
    loadButton.addActionListener(this);
    fileMenu.add(saveButton = new MenuItem("Save"));
saveButton.setEnabled(false);
    fileMenu.addSeparator();
    saveButton.addActionListener(this);
    fileMenu.add(exitButton = new MenuItem("Exit"));
    exitButton.addActionListener(this);
    mb.add(fileMenu);

// Construct the grail menu
    grailMenu = new Menu("Algorithm");
    grailMenu.add(exons[0].button);
    grailMenu.add(exons[1].button);
    grailMenu.add(exons[2].button);
    grailMenu.setEnabled(false);
    mb.add(grailMenu);

```

```

// Construct the windows menu
windowsMenu = new Menu("Window");
windowsMenu.add(seqWinButton= new MenuItem ("Sequence Window"));
seqWinButton.addActionListener(this);
windowsMenu.add(analWinButton= new MenuItem ("Analysis Window"));
analWinButton.addActionListener(this);
windowsMenu.add(feetWinButton= new MenuItem ("Features Window"));
feetWinButton.addActionListener(this);
windowsMenu.setEnabled(false);
mb.add(windowsMenu);

// Construct the features menu
featuresMenu = new Menu("Features");
featuresMenu.add(polya.button);
featuresMenu.add(promoter.button);
featuresMenu.add(cpg.button);
featuresMenu.add(rpt.button);
featuresMenu.add(gcButton = new CheckboxMenuItem("GC Content"));
featuresMenu.add(shadowExonsButton=new
    CheckboxMenuItem("Shadow Exons"));
shadowExonsButton.addItemListener(this);
featuresMenu.setEnabled(false);
mb.add(featuresMenu);
gcButton.addItemListener(this);

// Construct the assemble menu
assembleMenu = new Menu("Assemble");
assembleMenu.add(geneModels.button);
assembleMenu.setEnabled(false);
mb.add(assembleMenu);

// Set the look of the window
setMenuBar(mb);
setLayout(new BorderLayout());

// Make the main drawing canvas
mainCanvas = new MainCanvas(this);

// Make the panel that will hold the zoom bar
Panel zoombarPanel = new Panel();
zoombarPanel.setLayout(new FlowLayout());
zoombarPanel.add(new Label("Zoom "));
zoombarPanel.add(mainCanvas.zoombar);
zoombarPanel.add(mainCanvas.zoomValue);

```

```

// Make the status bar panel
    Panel statusBarPanel = new Panel();
    statusBarPanel.setLayout(new BorderLayout());
    statusBarPanel.add("West",mainCanvas.assemblyPanel);
    statusBarPanel.add("East",zoombarPanel);

// Make the bottom panel
    Panel bottomPanel = new Panel();
    bottomPanel.setLayout(new BorderLayout());
    bottomPanel.add("North",statusBarPanel);
    bottomPanel.add("South",mainCanvas.scrollbar);

// Make the panel that will hold the left position labels
    Panel leftLabelPanel = new Panel();
    leftLabelPanel.setLayout(new BorderLayout());
    leftLabelPanel.setBackground(Color.black);
    leftLabelPanel.setForeground(Color.white);
    leftLabelPanel.setFont(new Font("Courier",Font.BOLD,14));
    leftLabelPanel.add("North",mainCanvas.forwardLeft);
    leftLabelPanel.add("Center",new Label(" ",Label.CENTER));
    leftLabelPanel.add("South",mainCanvas.revLeft);

// Make the panel that will hold the right position labels
    Panel rightLabelPanel = new Panel();
    rightLabelPanel.setLayout(new BorderLayout());
    rightLabelPanel.setBackground(Color.black);
    rightLabelPanel.setForeground(Color.white);
    rightLabelPanel.setFont(new Font("Courier",Font.BOLD,14));
    rightLabelPanel.add("North",mainCanvas.forwardRight);
    rightLabelPanel.add("Center",new Label(" ",Label.CENTER));
    rightLabelPanel.add("South",mainCanvas.revRight);

// And dump it all in the main window
    add("Center",mainCanvas);
    add("West",leftLabelPanel);
    add("East",rightLabelPanel);
    add("South",bottomPanel);

// Make the analysisWindow
    analysisWindow = new AnalysisWindow(this);
    analysisWindow.addWindowListener(analysisWindow);

// Make the sequenceWindow

```

```

        sequenceWindow = new SequenceWindow(this);
        sequenceWindow.addWindowListener(sequenceWindow);

// Make the featuresWindow
        featuresWindow = new FeaturesWindow(this);
        featuresWindow.addWindowListener(featuresWindow);
    }

    int getAssemblyStart ()
    {
        int start;

        try {
            start = Integer.parseInt(mainCanvas.asmbMin.getText());
            if (start < 1) start = 1;
        }
        catch (NumberFormatException e)
        {
            start = 1;
        }

        mainCanvas.asmbMin.setText(Integer.toString(start));

        return start;
    }

    int getAssemblyEnd ()
    {
        int end;

        try {
            end = Integer.parseInt(mainCanvas.asmbMax.getText());
            if ((end <= getAssemblyStart()) || (end > sequenceLength))
                end = sequenceLength;
        }
        catch (NumberFormatException e)
        {
            end = sequenceLength;
        }

        mainCanvas.asmbMax.setText(Integer.toString(end));

        return end;
    }

```

```

static int invertBaseNumber (int p)
{
    return (sequenceLength+1-p);
}

public boolean eventNeedsAThread (AWTEvent evt)
{
    return (evt.getID() == WindowEvent.WINDOW_CLOSING);
}

public void threadedHandleEvent (AWTEvent evt)
{
    if (evt.getID() == WindowEvent.WINDOW_CLOSING) destroy();
}

public boolean myHandleEvent (AWTEvent evt) throws RuntimeException
{
    switch (evt.getID())
    {
        case AdjustmentEvent.UNIT_INCREMENT:
        case AdjustmentEvent.UNIT_DECREMENT:
        case AdjustmentEvent.BLOCK_INCREMENT:
        case AdjustmentEvent.BLOCK_DECREMENT:
        case AdjustmentEvent.TRACK:
            mainCanvas.repaint();
            return true;
        case ExceptionEvent.ERROR:

            Exception e = ((ExceptionEvent)evt).arg;
            throw new RuntimeException(e,
                "Unknown error: "+e.toString(),20,10);
        default: break;
    }
    return false;
}

public void setEnabled (boolean enabvar)
{
    if (enabvar)
    {
        super.setEnabled(true);
        analysisWindow.setEnabled(true);
        featuresWindow.setEnabled(true);
        sequenceWindow.setEnabled(true);
    }
}

```

```

    }

    else

    {
        super.setEnabled(false);
        analysisWindow.setEnabled(false);
        featuresWindow.setEnabled(false);
        sequenceWindow.setEnabled(false);
    }
}

public boolean actionNeedsAThread (AWTEvent evt)
{
    return ((evt.getSource() instanceof MenuItem) &&
        (evt.getSource() != gcButton));
}

public void threadedAction (ActionEvent evt) throws ErrorException
{
    try {
        // Die if the license has expired
        if (licenseExpired == TimeStamp.doomsday())
        {
            throw new ErrorException(
                "Your License Period has expired",20,1);
        }

        // Handle the event
        switch(evt.getSource())
        {
            case loadButton: load(); break;

            case saveButton: save(); break;

            case shadowExonsButton: toggleShadowExons(); break;

            case exons[0].button: exons[0].pressButton();
                setVersion(0); break;

            case exons[1].button: exons[1].pressButton();
                setVersion(1); break;

            case exons[2].button: exons[2].pressButton();
                setVersion(2); break;
        }
    }
}

```

```

        case polya.button: polya.pressButton(); break;
        case promoter.button: promoter.pressButton(); break;
            case cpg.button: cpg.pressButton(); break;

        case rpt.button: rpt.pressButton(); break;

        case geneModels.button:
geneModels.pressButton(getAssemblyStart(),getAssemblyEnd()); break;

        case seqWinButton: sequenceWindow.show(); break;
        case analWinButton: analysisWindow.show(); break;
        case featWinbutton: featuresWindow.show(); break;
        case exitButton: destroy(); break;
    }

    }

    catch (Exception e)
    {
        if (!(e instanceof ErrorException)) {
            throw new ErrorException(e,
                "Error: " + e.toString(),20,2);
        }
        else throw ((ErrorException)e);
    }
}

public boolean myAction (ActionEvent evt) throws ErrorException
{
    try {
        if (evt.getSource() instanceof MenuItem) {
            // Handle the event
            if (evt.getSource() == gcButton) mainCanvas.repaint();
        }
    }

    catch (Exception e)
    {
        if (!(e instanceof ErrorException)) {
            throw new ErrorException(e,
                "Error: " + e.toString(),20,3);
        }
        else throw ((ErrorException)e);
    }
}

```

```

        return false;
    }

void toggleShadowExons ()
{
    int i;

    // Update the exon lists to reflect the change
    for (i=0; i<3; i++)
    {
        if (shadowExonsButton.getState()) exons[i].showShadowExons();
        else exons[i].hideShadowExons();
        exons[i].textDisplay.repaint();
        analysisWindow.setVersion();
    }

    // Update the main window drawing
    mainCanvas.repaint();

    // And make sure the sequence window knows about this event too
    sequenceWindow.sequenceCanvas.repaint();
}

void setVersion (int v)
{
    int i;
    CheckboxMenuItem m;

    // Set the menu buttons accordingly
    for (i=0; i<3; i++)
    {
        m = (CheckboxMenuItem) grailMenu.getItem(i);
        m.setState(i == v);
    }

    // Change the title of the grail menu
    ver = v;

    // Enable / disable the assembly options as appropriate
    if (ver != 2)
    {
        mainCanvas.assemblyPanel.setVisible(false);
        mainCanvas.assemblyPanel.getParent().validate();
        disableMenu(assemblyMenu.getLabel());
    }
}

```



```

    else
    {
        mainCanvas.assemblyPanel.setVisible(true);
        mainCanvas.assemblyPanel.getParent().validate();
        enableMenu(assemblyMenu.getLabel());
    }

// Tell the analysis window about this event
analysisWindow.setVersion();

// Redraw the Sequence Window to reflect change
sequenceWindow.sequenceCanvas.scrollbar.setValue(0);
sequenceWindow.sequenceCanvas.repaint();

// Redraw the main canvas to reflect this change
mainCanvas.scrollbar.setValue(0);
mainCanvas.zoombar.setValue(100);
mainCanvas.repaint();
}

boolean load () throws Exception
{
    ReadStream in = null;
    AppletServer appletServer = null;
    boolean retval = false;
    String filename = null;

// Save first if we haven't already
if (loadFilename != null) save();

    if (amApplet) {
        FileChoice fc = null;

// Create the applet server
        appletServer = new AppletServer(remoteHost, remotePort,
            forwarderHost, forwarderPort, fileHost, localPort);

// Try to create the fileChoice window
        try {
            fc = new FileChoice(FileChoice.LOAD, appletServer);
        }

// If we couldn't connect to the applet server, then
// dispose of the FileChoice window and rethrow the exception
        catch (Exception e) {

```

```

        // Dispose of the fileChoice window
        if (fc != null) fc.dispose();

        // And rethrow the exception
        throw(e);
    }

    // Show the fileChoice window - this call blocks until
    // they select a file or cancel
    fc.show();

    // Get the name of the file to load
    filename = fc.getSelection();

    // Dispose of the fileChoice window
    fc.dispose();

    // If they clicked cancel then close the appletserver and return
    if (filename == null) {
        appletServer.exit();
        return false;
    }

    // Open the selected file
    appletServer.open(filename,"r");

    // Get the input stream associated with the file
    in = appletServer.read2eof();
}

else {
    String dirname;
    String fileSeparator;

    // Get the file separator
    fileSeparator = File.separator;

    // Throw up a file File dialog window
    FileDialog fd = new FileDialog(this,"Load",FileDialog.LOAD);
    fd.show();

    // If the user did Cancel
    if ((fd.getDirectory() == null) && (fd.getFile() == null)) {
        return false;
    }
}

```

```

// If the user did not select a file then bail
else if ((fd.getDirectory() == null) || (fd.getFile() == null)) {
    throw new RuntimeException(
        "No file selected for loading",20,4);
}
else if (fd.getFile().equals("")) {
    throw new RuntimeException("No file selected for loading",
        20,5);
}

// Get the name of the file to load. Beware: MacOS, for one, does
// NOT append a file separator to the directory name, but most
// other archs do. Work around this as follows:
dirname = fd.getDirectory();
if ((!dirname.endsWith(fileSeparator)) &&
    (!dirname.endsWith("/") &&
    (!dirname.endsWith("\\"))))
    filename = dirname + fileSeparator + fd.getFile();
else filename = dirname + fd.getFile();

// Try and open the file
try {
    in = new ReadStream(new FileInputStream(filename));
}
catch (FileNotFoundException e) {
    throw new RuntimeException(
        "Could not open \"" + filename + "\"",20,6);
}

int i;
String buf;
StringBuffer seq = new StringBuffer();

// Read in the first non-blank line
buf = in.readFirstNonBlankLine();

// If a name exists read it in
if (buf.charAt(0) == '>')
{
    // Skip leading spaces
    for (i=1; Character.isWhitespace(buf.charAt(i)); i++);
    seqname = buf.substring(i);

// Read in the next line
if ((buf = in.readLine()) == null)

```

```

        throw new Exception("Unexpected end of file",20,7);
    }

    // If a name does not exist set it to "Unnamed"
    else seqname = "Unnamed";

    // Read in the main sequence
    while (buf != null)
    {
        // Break if we've hit the start of the status report
        if (buf.equals(fileHeader)) break;

        // Convert the string to upper case
        buf = buf.toUpperCase();

        // Process character by character
        for (i=0; i<buf.length(); i++)
        {
            // Parse out spaces and digits
            if (Character.isWhitespace(buf.charAt(i)) ||
                Character.isDigit(buf.charAt(i)))
                continue;

            // Add remaining characters to the sequence string
            switch (buf.charAt(i))
            {
                case 'A': seq.append('A'); break;
                case 'C': seq.append('C'); break;
                case 'G': seq.append('G'); break;
                case 'T':
                case 'U': seq.append('T'); break;
                default: seq.append('C');
            }
        }

        // Read in the next line
        buf = in.readLine();
    }

    // Ensure the sequence is at least 100 bases in length
    if (seq.length() < 100)
        throw new Exception("Sequence too short",20,8);

    // Copy the sequence into permanent storage
    sequence = seq.toString();
    rsequence = StringOp.reverseSequence(sequence);

```

```

sequenceLength = sequence.length();
loadFilename = filename;

// Compute the GC content of the sequence
int numGCbands = 1 + (sequenceLength-
GC_WINDOW_SIZE)/GC_SHIFT_SIZE;
GC = new Color[numGCbands];
int[] gc = new int[numGCbands];
int b,g,j,numOverlapping = GC_WINDOW_SIZE/GC_SHIFT_SIZE;
for (b=g=0; (b < sequenceLength) && (g < gc.length);)
{
    if ((sequence.charAt(b)=='G') || (sequence.charAt(b)=='C'))
        for (j=0; j<numOverlapping; j++)
            if ((g+j < gc.length) && (b >= j*GC_SHIFT_SIZE))
                gc[g+j]++;

    if ((++b%GC_SHIFT_SIZE)==0)
    {
        gc[g] = (gc[g]*255)/GC_WINDOW_SIZE;
        GC[g] = new Color(gc[g],gc[g],gc[g]);
        g++;
    }
}

// Reset all the analysis options
saveButton.setEnabled(true); exons[0].reset();
exons[1].reset(); exons[2].reset();
polya.reset(); promoter.reset();
cpg.reset(); rpt.reset();
geneModels.reset();
shadowExonsButton.setState(false);

// Set window title to reflect new sequence
setTitle(title + " Sequence: " + seqname +
" Length: " + Integer.toString(sequence.length()));

// Hide the other windows
analysisWindow.setVisible(false);
sequenceWindow.setVisible(false);
featuresWindow.setVisible(false);

// If this is a status file then read in the Grail objects
if (buf != null)
{
    buf = in.readPrefix("[Comp = ");
    compMode = fileCompMode =

```

```

        Boolean.valueOf(buf.substring(0,buf.length()-1)).
        booleanValue();
    exons[0].load(in);
    exons[1].load(in);
    exons[2].load(in);
    polyA.load(in);
    cpG.load(in);
    promoter.load(in);
    rpt.load(in);
    geneModels.load(in);
        compMode = false;
}

// Read in the rest of the file just to be tidy - this is
// required if we are using the applet server.
while (buf != null) buf = in.readLine();

// close the input stream
if (amApplet) {
    appletServer.close();
    appletServer.exit();
}
else {
    in.close();
}

// Perform the analysis (if needed) and set the return value
retVal = exons[DEFAULT_VERSION].pressButton();
setVersion(DEFAULT_VERSION);

// Redisplay the other windows
sequenceWindow.reshape();
analysisWindow.reshape();
featuresWindow.reshape();
mainCanvas.reshape();

// Enable all the menu options
grailMenu.setEnabled(true);
windowsMenu.setEnabled(true);
featuresMenu.setEnabled(true);
assembleMenu.setEnabled(true);

// Finally, run the garbage collector and return
System.gc();
return retVal;
}

```

```

void save () throws Exception
{
    int i;
    WriteStream out = null;
    AppletServer appletServer = null;

    // Make the status filename by appending ".jgr" to sequence filename
    int dotPos = loadFilename.lastIndexOf(".");
    if (dotPos == -1) dotPos = loadFilename.length();
    saveFilename = loadFilename.substring(0,dotPos) + ".jgr";

    // Throw up a dialog box to let them change the filename
    SaveChoice sc = new SaveChoice(saveFilename);
    sc.show();
    saveFilename = sc.getSelection();
    sc.dispose();

    // If they cancelled then return immediately
    if (saveFilename == null) return;

    // Open the file
    if (amApplet) {
        // Create the applet server
        appletServer = new AppletServer(remoteHost, remotePort,
            forwarderHost, forwarderPort, fileHost, localPort);

        // Open the selected file
        appletServer.open(saveFilename,"w");

        // Get the output stream associated with the file
        out = appletServer.write2eof();
    }

    else {
        try {
            out = new WriteStream(new FileOutputStream(saveFilename));
        }
        catch (IOException e) {
            throw new RuntimeException(e,"Error writing file \"" +
                saveFilename + "\"",20,9);
        }
    }

    // Write out the sequence name
    out.writeBytes("> " + seqname + "\n\n");
}

```

```

// Write out the sequence string
    for (i=0; i<sequenceLength; i+=50)
    {
        out.writeBytes(sequence.substring(i,Math.min(i+50,
            sequenceLength)) + "\n");
    }
    out.writeBytes("\n");

// Write out the file header
    out.writeBytes(fileHeader + "\n");
    out.writeBytes("[Comp = " + fileCompMode + "]\n\n");

// Save all the Grail objects
    exons[0].save(out);
    exons[1].save(out);
    exons[2].save(out);
    polyA.save(out);
    cpg.save(out);
    promoter.save(out);
    rpt.save(out);
    geneModels.save(out);

// And write out a closing block just to be tidy
    out.writeBytes("[End Grail]\n");

// Close the file
    if (amApplet) {
        appletServer.endWrite2eof();
        appletServer.close();
        appletServer.exit();
    }

    else {
        out.close();
    }
}

////////////////////////////////////
// isApplet ()
//
// Returns true if we're running as an applet
////////////////////////////////////
    static boolean isApplet () { return amApplet; }

////////////////////////////////////

```



```

// start ()
//
// Called each time the applet web page is revisited
////////////////////////////////////
void start ()
{
    pack();
    show();
}

////////////////////////////////////
// stop ()
//
// Called each time the applet browser leaves the applet page
////////////////////////////////////
void stop () { }

////////////////////////////////////
// destroy ()
//
// Called each time the applet is unloaded
////////////////////////////////////
void destroy ()
{
    // Save the file before exiting - ignore any errors
    if (loadFilename != null) {
        try { save(); }
        catch (ErrorException e) {}
    }

    // If we're an applet dispose of all the windows
    if (amApplet) {
        applet.dispose();
        analysisWindow.dispose();
        sequenceWindow.dispose();
        featuresWindow.dispose();
        dispose();
    }

    // If we're an application exit the runtime environment
    else Runtime.getRuntime().exit(0);
}
}

```

```

//////////
// GraillItem.java
//////////

```

```

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.NoSuchElementException;

```

```

abstract class GraillItem

```

```

{
    char dirn;
    int start, end;
    boolean solid, selected=false;
    Color color, highlightColor;
    static Rectangle r = new Rectangle();

    String str2str (String s, int width)
    {
        while (s.length() < width) s = " " + s;
        return s;
    }

    String int2str (int x, int width)
    {
        String s = Integer.toString(x);
        while (s.length() < width) s = " " + s;
        return s;
    }

    String float2str (float x, int width, int dp)
    {
        String s = Float.toString(x);
        int dotpos = s.indexOf('.');
        if (dotpos == -1) s = s + ".00";
        else if (s.length() - 1 - dotpos < dp) {
            while (s.length()-1-s.indexOf('.') < dp) s = s + "0";
        }
        else s = s.substring(0,dotpos+dp+1);
    }
}

```

```

        while (s.length() < width) s = " " + s;
        return s;
    }

    int forwardStartBase ()
    {
        return (dim == 'f') ? start : Grail.invertBaseNumber(end);
    }

    boolean mouseClicked (int base, char[] strand)
    {
        if (dim != strand[0]) return false;
        if (dim == 'r') base = Grail.invertBaseNumber(base);
        return ((base >= start) && (base <= end));
    }

    void select () { selected = true; }

    void deselect () { selected = false; }

    void draw (Rectangle t, float basesPerPixel,
        Graphics g, Rectangle clipRect)
    // t.x = y position for reverse strand plots
    // t.y = y position for forward strand plots
    // t.width = X offset (i.e base number plotted at pixel x=0)
    // t.height = height of bar
    {
        if (dim == 'r') {
            r.y = t.x;
            r.x = (int)((Grail.invertBaseNumber(end)-t.width) / basesPerPixel);
        }
        else {
            r.y = t.y;
            r.x = (int)((start - t.width) / basesPerPixel);
        }
        r.width = Math.max((int)(Math.abs(1+end - start) / basesPerPixel),1);
        if (r.intersects(clipRect))
        {
            // Draw the bar
            g.setColor(color);
            if (solid) g.fillRect(r.x,r.y,r.width,t.height);
            else g.drawRect(r.x,r.y,r.width,t.height);

            // Highlight the bar if appropriate
            if (selected) {
                g.setColor(highlightColor);
            }
        }
    }

```

```

        g.drawRect(r.x-1,r.y-1,r.width+2,t.height+2);
    }
}

abstract GraillItem newItem ();

abstract String getLabel ();

abstract String getIdentifier ();

abstract String getRequestMessage ();

abstract String getFieldNames ();

abstract void write (WriteStream out) throws Exception;

abstract void read (ReadStream in) throws Exception;
}

```

```

////////////////////
// GrailObject.java
//
// Next available error code: 35-3
////////////////////

```

```

import java.io.*;
import java.awt.*;
import java.net.*;
import java.util.Vector;
import java.util.NoSuchElementException;

```

```

class GrailObject
{
    final String initMessage = "[Num = ";
    GrailItem itemClass;

    String statusString, startMessage, endMessage, requestMessage;
    MenuItem button;
    boolean gotten, showing, canGrow;
    Panel textDisplay;
    List itemList;

    int num;
    Vector item = new Vector();

    GrailObject (GrailItem sampleItem, boolean grows)
    {
        String identifier;

        growable(grows);
        button = growable() ?
            new MenuItem(sampleItem.getLabel()) :
            new CheckboxMenuItem(sampleItem.getLabel());
        itemList = new List(6, false);
        textDisplay = new Panel();
        textDisplay.setLayout(new BorderLayout());
        textDisplay.setFont(new Font("Courier", Font.PLAIN, 14));
        textDisplay.add("North", new Label(sampleItem.getFieldNames()));
        textDisplay.add("Center", itemList);
        identifier = sampleItem.getIdentifier();
        startMessage = "[Start " + identifier + "]";
        endMessage = "[End " + identifier + "]";
    }
}

```

```

        statusString = "[" + identifier + " = ";
        requestMessage = sampleItem.getRequestMessage();
        itemClass = sampleItem;
        got(false);
        hide();
    }

    GrailObject (GrailItem sampleItem)
    {
        this(sampleItem,false);
    }

    boolean got () { return gotten; }

    void got (boolean g) { gotten = g; }

    boolean growable () { return canGrow; }

    void growable (boolean g) { canGrow = g; }

    boolean isShowing () { return showing; }

    GrailItem getCurrentItem ()
    {
        int currentModel = itemList.getSelectedIndex();

        if (currentModel == -1) return (GrailItem) null;
        else return (GrailItem)(itemList.elementAt(currentModel));
    }

    void show ()
    {
        if (!growable()) {
            ((CheckboxMenuItem)button).setState(showing=true);
            itemList.setEnabled(true);
        }
    }

    void hide ()
    {
        if (!growable()) {
            ((CheckboxMenuItem)button).setState(showing=false);
            itemList.setEnabled(false);
        }
    }
}

```

```
void toggle () { if (isShowing()) hide(); else show(); }
```

```
void reset ()  
{  
    if (item.size() > 0) {  
        item.removeAllElements();  
        itemList.removeAll();  
    }  
    num = item.size();  
    got(false);  
    hide();  
}
```

```
boolean pressButton () throws Exception  
{  
    boolean retval;  
  
    if (growable()) showing = true; else toggle();  
    if (isShowing() && !got()) retval = analyze();  
    else retval = true;  
    Grail.mainCanvas.repaint();  
    Grail.sequenceWindow.sequenceCanvas.repaint();  
    return retval;  
}
```

```
void deselectAll ()  
{  
    int j;  
  
    for (j=0; j < item.size(); j++)  
        ((GrailItem)item.elementAt(j)).deselect();  
}
```

```
void select ()  
{  
    int j;  
  
    for (j=0; j < item.size(); j++)  
        ((GrailItem)item.elementAt(j)).deselect();  
    ((GrailItem)item.elementAt(itemList.getSelectedIndex())).select();  
}
```

```
boolean mouseClick (int base, char[] strand)  
{  
    int i,j;  
    GrailItem g;
```

```

for (i=0; i < item.size(); i++)
{
    if ((g=(GraillItem)item.elementAt(i)).mouseClick(base,strand))
    {
        if (itemList.isIndexSelected(i)) {
            itemList.deselect(i);
            g.selected = false;
        }
        else {
            // Deselect the current item (if one is selected.)
            if ((j = itemList.getSelectedIndex()) != -1) {
                ((GraillItem)item.elementAt(j)).selected = false;
                itemList.deselect(j);
            }
            // Select the new item
            itemList.select(i);

            // Make the newly selected item visible in the list
            // List.makeVisible() just plain doesn't work under
            // MacOS, so special-case it.
            if (System.getProperty("os.name").toUpperCase().
                indexOf("MACOS") == -1)
                itemList.makeVisible(i);

            // Let the item itself know that it is selected
            g.selected = true;
        }

        return true;
    }
}

return false;
}

void draw (Rectangle r, float basesPerPixel,
           Graphics g, Rectangle clipRect)
// r.x = y position for reverse strand plots
// r.y = y position for forward strand plots
// r.width = X offset (i.e base number plotted at pixel x=0)
// r.height = height of bar
{
    int i;

    for (i=0; i<item.size(); i++)

```



```

        ((GrailItem)item.elementAt(i)).draw(r,basesPerPixel,g,clipRect);
    }

void read (ReadStream in) throws RecognitionException
{
    String buf;
    int i;

    // Read in the start message
    in.readMessage(startMessage);

    try {
        // Read in the [Num = n] line
        buf = in.readPrefix(initMessage);

        // Parse out the number of entries to be read
        num = Integer.parseInt(buf.substring(0,buf.length()-1));

        // Read in the items
        for (i=0; i<num; i++)
        {
            item.addElement(itemClass.newItem());
            ((GrailItem)item.lastElement()).read(in);
            itemList.addItem(((GrailItem)item.lastElement()).toString());
        }

        // update the number of items in the collection
        num = item.size();

        // Read in the end message
        in.readMessage(endMessage);
    }

    catch (NumberFormatException e) {
        throw new RecognitionException(e,"Error reading data",35,1);
    }

    catch (NoSuchElementException e) {
        throw new RecognitionException(e,"Error reading data",35,2);
    }

    // Run the garbage collector to pick up any trash we left lying around
    System.gc();
}

void load (ReadStream in) throws RecognitionException

```

```

{
// Read in the object status line
String buf = in.readPrefix(statusString);

// Parse out the status
Boolean status = Boolean.valueOf(buf.substring(0,buf.length()-1));

// If this object is present then load it in
if (status.booleanValue())
{
    read(in);
    got(!growable());
}
}

void save (WriteStream out) throws Exception
{
    boolean status = growable() ? (item.size() > 0) : got();

// Write out the status of the object
    out.writeBytes(statusString + status + "]\n");

// And save if there is anything to save
    if (status) write(out);
}

void write (WriteStream out) throws Exception
{
    int i;

// Write out the start message
    out.writeBytes(startMessage + "\n");

// Write out the [Num = n] line
    out.writeBytes(initMessage + item.size() + "]\n");

// Write out the items
    for (i=0; i<item.size(); i++)
    {
        ((GraillItem)item.elementAt(i)).write(out);
    }

// Write out the end message
    out.writeBytes(endMessage + "\n\n");
}

```

```

boolean analyze () throws Exception
{
    boolean retval = false;

    // Applets use the applet server to contact the main server
    if (Grail.isApplet()) {

        // Make an applet server
        AppletServer appletServer = new AppletServer(Grail.remoteHost,
            Grail.remotePort, Grail.forwarderHost,
            Grail.forwarderPort, Grail.fileHost, Grail.localPort);

        // Send the request and get the stream through which the
        // reply will be sent
        ReadStream in = appletServer.sendRequest(
            requestMessage, Grail.sequence, endMessage);

        // Read the feature from the input stream
        read(in);

        // Shut down the applet server
        appletServer.exit();

        // everything worked so return true
        got(!growable());
        retval = true;
    }

    // Applications connect to the main server directly
    else {
        try {
            // connect to the server
            Socket connection = new
                Socket(Grail.remoteHost, Grail.remotePort);

            // open up the communication streams
            ReadStream in = new ReadStream(new DataInputStream(
                connection.getInputStream()));
            WriteStream out = new WriteStream(new DataOutputStream(
                connection.getOutputStream()));

            // send the request
            out.writeBytes(requestMessage + Grail.sequence + ".");

            // catch the response
            read(in);

```

```

// close the socket (and streams)
in.close();
out.close();
connection.close();

// everything worked so return true
got(!growable());
retval = true;
}

catch (UnknownHostException e)
{
    throw new ErrorException(e,
        "Unknown host: \"" + Grail.remoteHost + "\"",2,1);
}

catch (IOException e)
{
    throw new ErrorException(e,
        "IO Exception: " + e.toString(),2,2);
}
}

return retval;
}
}

```

```

//////////
// MainCanvas
//////////

import java.awt.*;
import java.awt.AWTEvent;

class MainCanvas extends Canvas
{
    static final int ERROR = -1;

// MyScrollbar scrollbar, zoombar;
    Scrollbar scrollbar, zoombar;
    MyLabel forwardLeft, forwardRight, revLeft, revRight;
    TextField asmbMin, asmbMax;
    int fmin,fmax,rmin,rmax,mid;
    Panel assemblyPanel;
    Label zoomValue;
    Grail myParent;

    int maxExonScore = 80;
    int yScaleFactor = 2;
    int xScaleFactor = 10;

    int barSep = 3;
    int gcBarHeight = 10;
    int rptBarHeight = 5;
    int promBarHeight = 5;
    int cpgBarHeight = gcBarHeight + 1;
    int exonBarHeight = 5;

    int gcY = gcBarHeight/2;
    int cpgY = gcY + 1;
    int botPolyaY = cpgY+1;
    int botRptY = cpgY + barSep;
    int topRptY = botRptY + rptBarHeight;
    int botPromY = topRptY + barSep;
    int topPromY = botPromY + promBarHeight;
    int baseline = topPromY + barSep;
    int topPolyaY = baseline-1;
    int topline = baseline + maxExonScore / yScaleFactor;
    int botExonY = topline + barSep;
    int topExonY = botExonY + exonBarHeight;

```

```

int botShadowY = topExonY + barSep;
int topShadowY = botShadowY + exonBarHeight;
int botModelY = topShadowY + barSep;
int topModelY = botModelY + exonBarHeight;

int polyaBarHeight = topPolyaY-botPolyaY+1;

// Graphics buffer and size stuff
int bufferWidth, bufferHeight;
Dimension prefSize, minSize;
Rectangle clipRect;
Image bufferImage;
Graphics buffer;

MainCanvas (Grail g)
{
// Save pointer to the parent
myParent = g;

// Set the background color and font
setBackground(Color.black);
setFont(new Font("Courier",Font.PLAIN,14));

// Set the minimum and preferred dimensions. The window takes up
// 128 pixels plus the width of this canvas.
int screenWidth = Toolkit.getDefaultToolkit().getScreenSize().width;
prefSize = new Dimension(screenWidth-160,212);
minSize = new Dimension(screenWidth-160,212);

// Set up the position labels and zoom bar value label
forwardLeft = new MyLabel(6,Label.LEFT); forwardLeft.setVisible(false);
forwardRight = new MyLabel(6,Label.RIGHT); forwardRight.setVisible(false);
revLeft = new MyLabel(6,Label.LEFT); revLeft.setVisible(false);
revRight = new MyLabel(6,Label.RIGHT); revRight.setVisible(false);
zoomValue = new Label("x10",Label.CENTER);

// Set up the scroll bars
scrollbar = new Scrollbar(Scrollbar.HORIZONTAL);
scrollbar.addAdjustmentListener(grail);
scrollbar.addAdjustmentListener(sequenceWindow);
zoombar = new Scrollbar(Scrollbar.HORIZONTAL,100,10,10,100);
zoombar.addAdjustmentListener(grail);
zoombar.addAdjustmentListener(sequenceWindow);
zoombar.setUnitIncrement(10);
scrollbar.setEnabled(false);
zoombar.setEnabled(false);

```

```

// Set up the gene assembly range markers
asemblMin = new TextField("0000000",6);
asemblMax = new TextField("0000000",6);

// Make the panel that displays assembly range
assemblyPanel = new Panel();
assemblyPanel.setLayout(new FlowLayout());
assemblyPanel.add(new Label("Assemble from "));
assemblyPanel.add(asemblMin);
assemblyPanel.add(new Label(" to "));
assemblyPanel.add(asemblMax);
assemblyPanel.setVisible(false);
}

public Dimension getPreferredSize ()
{
    return prefSize;
}

public Dimension getMinimumSize ()
{
    return minSize;
}

void reshow ()
{
    forwardLeft.setVisible(true); forwardRight.setVisible(true);
    revLeft.setVisible(true); revRight.setVisible(true);
    forwardLeft.getParent().validate();
    forwardRight.getParent().validate();
    assemblyPanel.setVisible(true); assemblyPanel.getParent().validate();
    scrollbar.setEnabled(true);
    zoombar.setEnabled(true);
    asemblMin.setText("1");
    asemblMax.setText(Integer.toString(Grail.sequenceLength));
    repaint();
}

void jumpToBase (int base)
{
    scrollbar.setValue(base-(fmax-fmin)/2);
    repaint();
}

public void paint (Graphics g)

```

```

{
    update(g);
}

public void update (Graphics g)
{
    if (!zoombar.isEnabled()) return;

    try {
        // If the buffer does not exist, or is a different size than
        // the current canvas, make a new one.
        if ((buffer == null) || (bufferWidth != getSize().width) ||
            (bufferHeight != getSize().height)) {
            bufferWidth = getSize().width;
            bufferHeight = getSize().height;
            bufferImage = createImage(bufferWidth,bufferHeight);
            buffer = bufferImage.getGraphics();

            // Get the clipRect - images don't have one it appears
            clipRect = new Rectangle(0,0,bufferWidth,bufferHeight);
        }

        // Blank out the buffer
        buffer.setColor(getBackground());
        buffer.fillRect(0,0,bufferWidth,bufferHeight);
        buffer.setColor(getForeground());

        int page, zoom = zoombar.getValue();
        Rectangle r = new Rectangle(0,0,bufferWidth,bufferHeight);
        Rectangle p = new Rectangle();
        int gcnt,xoffset,maxx,y;
        double x1,x2,width;

        // Get the midpoint of the window
        mid = r.height/2;

        // Update the scrollbar limits
        scrollbar.setValues(scrollbar.getValue(),page=(int)(r.width*
            xScaleFactor*(100.0/zoom)),0,Grail.sequenceLength-page);
        scrollbar.setUnitIncrement((int)(page/10.0));

        // Update the position labels
        forwardLeft.setText(Integer.toString(fmin=scrollbar.getValue()+1));
        forwardRight.setText(Integer.toString(fmax=Math.min(fmin+page-1,
            Grail.sequenceLength)));
        revLeft.setText(Integer.toString(rmax=Grail.

```



```

        invertBaseNumber(fmin)));
    revRight.setText(Integer.toString(rmin=Grail.
        invertBaseNumber(fmax)));

// Update the zoom bar value
zoomValue.setText("x" + (zoom/10));

// Draw the exons
g.setColor(Color.white);
r.x = mid + botExonY;
r.y = mid - topExonY;
r.width = (xoffset = scrollbar.getValue());
r.height = exonBarHeight;
p.x = mid - baseline;
p.y = mid - topline;
p.width = mid + baseline;
p.height = mid + topline;
Grail.exons[Grail.ver].draw(r,p,mid-topShadowY,mid+botShadowY,
    yScaleFactor,(float)(xScaleFactor*100.0/zoom),buffer,clipRect);

// Draw the GC content if appropriate
if (Grail.gcButton.getState()) {
    if ((gcnt=(xoffset-(Grail.GC_WINDOW_SIZE/2)) /
        Grail.GC_SHIFT_SIZE) < 0) gcnt = 0;
    maxx = clipRect.x + clipRect.width;
    width = Grail.GC_SHIFT_SIZE/(xScaleFactor*100.0/zoom);
    y = mid - gcY;
    x1 = (10*(gcnt+2)-xoffset)/(xScaleFactor*100.0/zoom);
    x2 = x1 + 10/(xScaleFactor*100.0/zoom);
    for (; ((x1 < maxx) && (gcnt < Grail.GC.length));
        x1=x2,x2+=width,gcnt++) {
        buffer.setColor(Grail.GC[gcnt]);
        buffer.fillRect((int)x1,y,(int)x2-(int)x1,gcBarHeight);
    }
}

// Draw the model exons, if appropriate
if (Grail.assembleMenu.isEnabled()) {
    r.x = mid + botModelY;
    r.y = mid - topModelY;
    r.width = xoffset;
    r.height = exonBarHeight;
    Grail.geneModels.draw(r,(float)(xScaleFactor*100.0/zoom),
        buffer,clipRect);
}

```

```

// Draw the Pol II Promoter sites
if (Grail.promoter.isShowing()) {
    r.x = mid + botPromY - 1; // y position for reverse strand
    r.y = mid - topPromY;     // y position for forward strand
    r.width = xoffset;        // X offset
    r.height = promBarHeight; // height of bar
    Grail.promoter.draw(r,(float)(xScaleFactor*100.0/zoom),
        buffer,clipRect);
}

// Draw the CpG Islands
if (Grail.cpg.isShowing()) {
    r.x = r.y = mid - cpgY;
    r.width = xoffset;
    r.height = cpgBarHeight;
    Grail.cpg.draw(r,(float)(xScaleFactor*100.0/zoom),
        buffer,clipRect);
}

// Draw the Rpt sites
if (Grail.rpt.isShowing()) {
    r.x = mid + botRptY - 1; // y position for reverse strand
    r.y = mid - topRptY;     // y position for forward strand
    r.width = xoffset;        // X offset
    r.height = rptBarHeight; // height of bar
    Grail.rpt.draw(r,(float)(xScaleFactor*100.0/zoom),
        buffer,clipRect);
}

// Draw the PolyA sites
if (Grail.polya.isShowing()) {
    r.x = mid + botPolyaY - 1; // y position for reverse strand
    r.y = mid - topPolyaY;     // y position for forward strand
    r.width = xoffset;        // X offset
    r.height = polyabarHeight; // height of bar
    Grail.polya.draw(r,(float)(xScaleFactor*100.0/zoom),
        buffer,clipRect);
}

// And paste the buffer onto the screen
g.drawImage(bufferImage,0,0,this);
}

// Pass any errors to the Main window to be dealt with cleanly
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

```

```

    }
}

public boolean handleEvent (Event evt)
{
    if (!zoombar.isEnabled()) return false;

    try {
        if (evt.id == Event.MOUSE_DOWN)
        {
            // Convert the X position to a forward strand base number
            int base = (int)(evt.x*xScaleFactor*100.0/zoombar.getValue() +
                scrollbar.getValue());

            // Convert the Y position to a from-midpoint distance
            int y = Math.abs(evt.y-mid);
            char[] strand = {'f','s'};
            if (evt.y > mid) strand[0] = 'r';

            // Hit the Polya sites, if appropriate
            Polya.clickWidth = (int)(2*xScaleFactor*100.0/zoombar.
                getValue());
            if (Grail.polya.isShowing() &&
                (y >= botPolyaY) && (y <= topPolyaY))
                if (Grail.polya.mouseClick(base,strand))
                {
                    if (Grail.polya.itemList.getSelectedIndex() != -1)
                        Grail.sequenceWindow.sequenceCanvas.
                            jumpToBase(base);
                    else Grail.sequenceWindow.sequenceCanvas.repaint();
                    repaint();
                    return true;
                }

            // Hit the Pol II promoters, if appropriate
            if (Grail.promoter.isShowing() &&
                (y >= botPromY) && (y <= topPromY))
                if (Grail.promoter.mouseClick(base,strand))
                {
                    if (Grail.promoter.itemList.getSelectedIndex() != -1)
                        Grail.sequenceWindow.sequenceCanvas.
                            jumpToBase(base);
                    else Grail.sequenceWindow.sequenceCanvas.repaint();
                    repaint();
                    return true;
                }
        }
    }
}

```

```

// Hit the Rpt's, if appropriate
if (Grail.rpt.isShowing() && (y >= botRptY) && (y <= topRptY))
    if (Grail.rpt.mouseClick(base,strand))
    {
        if (Grail.rpt.itemList.getSelectedIndex() != -1)
            Grail.sequenceWindow.sequenceCanvas.
                jumpToBase(base);
        else Grail.sequenceWindow.sequenceCanvas.repaint();
        repaint();
        return true;
    }

// Hit the standard Exons, if appropriate
if ((y >= botExonY) && (y <= topExonY))
    if (Grail.exons[Grail.ver].mouseClick(base,strand))
    {
        if (Grail.exons[Grail.ver].itemList.
            getSelectedIndex() != -1)
            Grail.sequenceWindow.sequenceCanvas.
                jumpToBase(base);
        else Grail.sequenceWindow.sequenceCanvas.repaint();
        repaint();
        return true;
    }

// Hit the alternate Exons, if appropriate
strand[1] = 'a';
if (Grail.exons[Grail.ver].showingShadows &&
    (y >= botShadowY) && (y <= topShadowY))
    if (Grail.exons[Grail.ver].mouseClick(base,strand))
    {
        if (Grail.exons[Grail.ver].itemList.
            getSelectedIndex() != -1)
            Grail.sequenceWindow.sequenceCanvas.
                jumpToBase(base);
        else Grail.sequenceWindow.sequenceCanvas.repaint();
        repaint();
        return true;
    }

// Hit the model exons if appropriate
if (Grail.assembleMenu.isEnabled() &&
    (y >= botModelY) && (y <= topModelY))
    if (Grail.geneModels.mouseClick(base,strand))
    {

```

```

        int m;
        if ((m=Grail.geneModels.itemList.
            getSelectedIndex()) != -1)
            if (((Gene_Model)Grail.geneModels.item.
                elementAt(m)).exons.itemList.
                    getSelectedIndex() != -1)
                Grail.sequenceWindow.sequenceCanvas.
                    jumpToBase(base);
            else Grail.sequenceWindow.sequenceCanvas.repaint();
        repaint();
        return true;
    }

    // Hit the CpG islands, if appropriate
    strand[0] = 'T';
    if (Grail.cpg.isShowing() && (y <= cpgY))
        if (Grail.cpg.mouseClick(base,strand))
        {
            if (Grail.cpg.itemList.getSelectedIndex() != -1)
                Grail.sequenceWindow.sequenceCanvas.
                    jumpToBase(base);
            else Grail.sequenceWindow.sequenceCanvas.repaint();
            repaint();
            return true;
        }
    }

    return false;
}

// Pass any errors to Grail to be dealt with there
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

return false;
}
}

```

```
////////////////////////////////////
```

```
// Model_Exon.java
```

```
//
```

```
// Next available error code: 32-3
```

```
////////////////////////////////////
```

```
import java.awt.Graphics;
import java.awt.Rectangle;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;
```

```
class Model_Exon extends GrailItem
```

```
{
    int frame;
    int score;
    Translation trans;
```

```
    static String buttonLabel, identifier, requestMessage, fieldNames;
```

```
    Model_Exon ()
```

```
    {
        solid = true;
        color = Grail.modelColor;
        highlightColor = Grail.modelHighlightColor;
        buttonLabel = "";
        identifier = "Model_Exons";
        requestMessage = "";
        fieldNames = "Frm    Position    Score";
        trans = new Translation();
    }
```

```
    GrailItem newItem () { return new Model_Exon(); }
```

```
    String getLabel () { return buttonLabel; }
```

```
    String getIdentifier () { return identifier; }
```

```
    String getRequestMessage () { return requestMessage; }
```

```
    String getFieldNames () { return fieldNames; }
```

```
    public String toString ()
    {
```

```

StringBuffer s = new StringBuffer(50);
int i; for (i=0; i<25; i++) s.append(' ');

try
{
    s.insert(1,frame);
    s.insert(5,start);
    s.insert(14,end);
    s.insert(23,score);
}
catch (ArrayIndexOutOfBoundsException e) {}
catch (StringIndexOutOfBoundsException e) {}

return s.toString().trim();
}

void draw (Rectangle t, int fth, int rth, float basesPerPixel,
Graphics g, Rectangle clipRect)
// t.x = y position for reverse strand plots
// t.y = y position for forward strand plots
// t.width = X offset (i.e base number plotted at pixel x=0)
// t.height = height of bar
// fth = forward strand translation height
// rth = reverse strand translation height
{
    super.draw(t,basesPerPixel,g,clipRect);
    if (selected) {
        g.setColor(color);
        trans.draw(t.width,fth,rth,dirn,basesPerPixel,g);
    }
}

void write (WriteStream out) throws Exception
{
    if ((!Grail.fileCompMode) || (dirn == 'f')) {
        out.writeBytes (int2str(start,7) + " " + int2str(end,7) + " " +
                        int2str(frame,3) + " " + int2str(score,6) + "\n");
    }
    else {
        out.writeBytes (int2str(1+Grail.sequenceLength-end,7) + " " +
                        int2str(1+Grail.sequenceLength-start,7) + " " +
                        int2str(frame,3) + " " + int2str(score,6) + "\n");
    }
}

void read (ReadStream in) throws Exception

```

```

{
    StringTokenizer st;

    // Read the first non-blank line
    st = new StringTokenizer(in.readFirstNonBlankLine());

    try {

        // Parse out the start position
        start = Integer.parseInt(st.nextToken());

        // Parse out the end position
        end = Integer.parseInt(st.nextToken());

        // Parse out the frame
        frame = Integer.parseInt(st.nextToken());

        // Grab the score
        score = Integer.parseInt(st.nextToken());

        //      Set the direction character
        dirn = Grail.modelDirn;

        //      Correct for complementary mode
        if ((Grail.compMode) && (dirn == 'r')) {
            int tmp;
            tmp = 1 + Grail.sequenceLength - end;
            end = 1 + Grail.sequenceLength - start;
            start = tmp;
        }

        // Update the translation to reflect this exon
        trans.start = start;
        trans.end = end+3;
    }

    catch (NumberFormatException e) {
        throw new RuntimeException(e, "Error reading model exon", 32, 1);
    }

    catch (NoSuchElementException e) {
        throw new RuntimeException(e, "Error reading model exon", 32, 2);
    }
}
}

```



```

////////////////////////////////////
// MyLabel.java
//
// On PCs you cannot set the foreground color of labels.
// This class overcomes that restriction.
////////////////////////////////////

import java.awt.*;

class MyLabel extends Canvas
{
    String text = "";
    Dimension minSize, prefSize;
    int x,y,charWidth,justification;

    MyLabel (int myWidth, int just)
    {
        justification = just;

        Font font = new Font("Courier",Font.BOLD,14);
        setFont(font);
        validate();

        minSize = new Dimension();
        prefSize = new Dimension();
        FontMetrics fm = getFontMetrics(font);
        charWidth = fm.charWidth('0');
        minSize.width = prefSize.width = charWidth * (myWidth+2);
        minSize.height = prefSize.height = fm.getHeight();
        x = charWidth;
        y = fm.getHeight()-fm.getDescent()-1;
        validate();
    }

    String getText () { return text; }

    void setText (String t)
    {
        text = t;

        if (t.length() > ((prefSize.width/charWidth)-2)) {
            minSize.width = prefSize.width = charWidth * (t.length()+2);
        }
    }
}

```

```

        repaint();
    }

    public void paint (Graphics g)
    {
        if (justification == Label.RIGHT)
            x = getSize().width - text.length()*charWidth - charWidth;
        g.drawString(text,x,y);
    }

    public Dimension getPreferredSize ()
    {
        return prefSize;
    }

    public Dimension getMinimumSize ()
    {
        return minSize;
    }
}

```

```
//////////  
// MyScrollbar.java  
//////////
```

```
import java.awt.Scrollbar;  
import java.awt.Event;  
import java.awt.event.*;
```

```
class MyScrollbar extends Scrollbar  
// Overrides the predefined scrollbar which does  
// not process line increment events correctly  
{  
    MyScrollbar ()  
    {  
        super();  
    }  
  
    MyScrollbar (int orientation)  
    {  
        super(orientation);  
    }  
  
    MyScrollbar (int orientation, int value, int visible, int min, int max)  
    {  
        super(orientation,value,visible,min,max);  
    }  
  
    public boolean handleEvent (Event evt)  
    {  
        MyScrollbar s = (MyScrollbar)(evt.target);  
  
        switch (evt.id)  
        {  
            case Event.SCROLL_LINE_UP:  
                s.setValue(s.getValue() + 1 - s.getUnitIncrement());  
                break;  
            case Event.SCROLL_LINE_DOWN:  
                s.setValue(s.getValue() + s.getUnitIncrement() - 1);  
                break;  
            case Event.SCROLL_PAGE_UP:  
                s.setValue(s.getValue() + 10 - s.getVisibleAmount());  
                break;  
            case Event.SCROLL_PAGE_DOWN:
```

```
        s.setValue(s.getValue() + s.getVisibleAmount() - 10);
    default:
        break;
}

return false;
}
```

```
////////////////////////////////////
```

```
// PolyA.java
```

```
//
```

```
// Next available error code: 25-4
```

```
////////////////////////////////////
```

```
import java.awt.Graphics;
import java.awt.Rectangle;
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;
```

```
class PolyA extends GrailItem
```

```
{
```

```
    float score;
```

```
    final int minDrawingWidth = 5;
```

```
    static int clickWidth = 0;
```

```
    static String buttonLabel, identifier, requestMessage, fieldNames;
```

```
    PolyA ()
```

```
    {
```

```
        solid = true;
```

```
        color = Grail.polyaColor;
```

```
        highlightColor = Grail.polyaHighlightColor;
```

```
        buttonLabel = "PolyA Site";
```

```
        identifier = "PolyA";
```

```
        requestMessage = "dp.";
```

```
        fieldNames = "Str Start End Score";
```

```
    }
```

```
    public String toString ()
```

```
    {
```

```
        StringBuffer s = new StringBuffer(100);
```

```
        int i; for (i=0; i<50; i++) s.append(' ');
```

```
        try
```

```
        {
```

```
            s.insert(1,dirn);
```

```
            s.insert(4,start);
```

```
            s.insert(13,end);
```

```

        s.insert(22,score);
    }
    catch (ArrayIndexOutOfBoundsException e) {}
    catch (StringIndexOutOfBoundsException e) {}

    return (" " + s.toString().trim());
}

GrailItem newItem () { return new Polya(); }

String getLabel () { return buttonLabel; }

String getIdentifier () { return identifier; }

String getRequestMessage () { return requestMessage; }

String getFieldNames () { return fieldNames; }

void draw (Rectangle t, float basesPerPixel,
    Graphics g, Rectangle clipRect)
// t.x = y position for reverse strand plots
// t.y = y position for forward strand plots
// t.width = X offset (i.e base number plotted at pixel x=0)
// t.height = height of bar
{
    int boxY=0;

    if (dirn == 'r') {
        r.y = t.x;
        r.x = (int)((Grail.invertBaseNumber(end)-t.width) / basesPerPixel);
    }
    else {
        r.y = t.y;
        r.x = (int)((start - t.width) / basesPerPixel);
    }
    r.width = Math.max((int)(Math.abs(1+end - start) / basesPerPixel),1);
    if (r.intersects(clipRect))
    {
        // Draw the bar
        g.setColor(color);
        g.fillRect(r.x,r.y,r.width,t.height);
        if (r.width < minDrawingWidth) {
            if (dirn == 'r') boxY = r.y + t.height - minDrawingWidth;
            else boxY = r.y;
            g.fillRect(r.x-minDrawingWidth/2,boxY,
                minDrawingWidth,minDrawingWidth);
        }
    }
}

```

```

    }

    // Highlight the bar if appropriate
    if (selected) {
        g.setColor(highlightColor);
        g.drawRect(r.x-1,r.y-1,r.width+2,t.height+2);
        if (r.width < minDrawingWidth)
            g.drawRect(r.x-minDrawingWidth/2-1,boxY-1,
                minDrawingWidth+2,minDrawingWidth+2);
    }
}

boolean mouseClicked (int base, char[] strand)
{
    if (dirn != strand[0]) return false;
    if (dirn == 'r') base = Grail.invertBaseNumber(base);
    return ((base >= start-clickWidth) && (base <= end+clickWidth));
}

void write (WriteStream out) throws Exception
{
    if ((!Grail.fileCompMode) || (dirn == 'f')) {
        out.writeBytes (dirn + " " + int2str(start,6) + " " +
            int2str(end,6) + " " + float2str(score,5,3) + "\n");
    }
    else {
        out.writeBytes (dirn + " " +
            int2str(1+Grail.sequenceLength-end,6) + " " +
            int2str(1+Grail.sequenceLength-start,6) + " " +
            float2str(score,5,3) + "\n");
    }
}

void read (ReadStream in) throws Exception
{
    String buf;
    StringTokenizer st;

    // Read the first non-blank line
    st = new StringTokenizer(in.readFirstNonBlankLine());

    try {

        // Grab the direction character
        if ((buf = st.nextToken()).length() != 1)

```

```

        throw new RuntimeException("Error reading PolyA site",25,1);
        dirn = buf.charAt(0);

// Parse out the start position
        start = Integer.parseInt(st.nextToken());

// Parse out the end position
        end = Integer.parseInt(st.nextToken());

// Grab the score
        score = Float.valueOf(st.nextToken()).floatValue();

        //      Correct for complementary mode
        if ((Grail.compMode) && (dirn == 'r')) {
            int tmp;
            tmp = 1 + Grail.sequenceLength - end;
            end = 1 + Grail.sequenceLength - start;
            start = tmp;
        }
    }

    catch (NumberFormatException e) {
        throw new RuntimeException("Error reading PolyA site",25,2);
    }

    catch (NoSuchElementException e) {
        throw new RuntimeException("Error reading PolyA site",25,3);
    }
}
}

```



```

////////////////////
// Prom.java
//
// Next available error code: 26-4
////////////////////

import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;

class Prom extends GrailItem
{
    float score;
    int CAAT, GCBox, TATA, Cap, ATG;

    static String buttonLabel, identifier, requestMessage, fieldNames;

    Prom ()
    {
        solid = true;
        color = Grail.promoterColor;
        highlightColor = Grail.promoterHighlightColor;
        buttonLabel = "Promoters";
        identifier = "polIIProm";
        requestMessage = "d2.";
        fieldNames = "Str Score CAAT GCBox TATA Cap ATG ";
    }

    public String toString ()
    {
        StringBuffer s = new StringBuffer(100);
        int i; for (i=0; i<50; i++) s.append(' ');

        try
        {
            s.insert(1,dirn);
            s.insert(4,score);
            s.insert(10,CAAT);
            s.insert(19,GCBox);
            s.insert(28,TATA);
            s.insert(37,Cap);

```

```

        s.insert(46,ATG);
    }
    catch (ArrayIndexOutOfBoundsException e) {}
    catch (StringIndexOutOfBoundsException e) {}

    return (" " + s.toString().trim());
}

GrailItem newItem () { return new Prom(); }

String getLabel () { return buttonLabel; }

String getIdentifier () { return identifier; }

String getRequestMessage () { return requestMessage; }

String getFieldNames () { return fieldNames; }

void write (WriteStream out) throws ErrorException
{
    if ((!Grail.fileCompMode) || (dirn == 'f')) {
        out.writeBytes (dirn + " " + float2str(score,5,3) + " " +
                        int2str(CAAT,6) + " " +
                        int2str(GCBox,6) + " " + int2str(TATA,6) + " " +
                        int2str(Cap,6) + " " + int2str(ATG,6) + "\n");
    }
    else {
        out.writeBytes (dirn + " " + float2str(score,5,3) + " " +
                        int2str(CAAT==1 ?-1:(1+Grail.sequenceLength-
CAAT),6) + " " +
                        int2str(GCBox==1 ?-1:(1+Grail.sequenceLength-GCBox),6) + " " +
                        int2str(TATA==1 ?-1:(1+Grail.sequenceLength-TATA),6)
+ " " +
                        int2str(Cap==1 ?-1:(1+Grail.sequenceLength-Cap),6) + " "
+
                        int2str(ATG==1 ?-1:(1+Grail.sequenceLength-ATG),6) +
"\n");
    }
}

void read (ReadStream in) throws ErrorException
{
    String buf;
    StringTokenizer st;

    // Read the first non-blank line

```

```

st = new StringTokenizer(in.readFirstNonBlankLine());

try {

// Grab the direction character
if ((buf = st.nextToken()).length() != 1)
    throw new RuntimeException("Error reading promoter site",26,1);
dirn = buf.charAt(0);

// Parse out the score
score = Float.valueOf(st.nextToken()).floatValue();

// Parse out the CAAT position
CAAT = Integer.parseInt(st.nextToken());

// Parse out the GCBox position
GCBox = Integer.parseInt(st.nextToken());

// Parse out the TATA position
TATA = Integer.parseInt(st.nextToken());

// Parse out the Cap position
Cap = Integer.parseInt(st.nextToken());

// Parse out the ATG position
ATG = Integer.parseInt(st.nextToken());

        //      Correct for complementary mode
        if ((Grail.compMode) && (dirn == 'r')) {
            if (CAAT != -1) CAAT = 1 + Grail.sequenceLength -
CAAT;
            if (GCBox != -1) GCBox = 1 + Grail.sequenceLength -
GCBox;
            if (TATA != -1) TATA = 1 + Grail.sequenceLength -
TATA;
            if (Cap != -1) Cap = 1 + Grail.sequenceLength - Cap;
            if (ATG != -1) ATG = 1 + Grail.sequenceLength - ATG;
        }

// Get the start position
if (CAAT != -1) start = CAAT;
else if (GCBox != -1) start = GCBox;
else if (TATA != -1) start = TATA;
else if (Cap != -1) start = Cap;
else start = ATG;

```

```

// Get the end position
if (ATG != -1) end = ATG+2;
else if (Cap != -1) end = Cap;
else if (TATA != -1) end = TATA;
else if (GCBox != -1) end = GCBox;
else end = CAAT;
}

catch (NumberFormatException e) {
    throw new RuntimeException(e,"Error reading promoter site",26,2);
}

catch (NoSuchElementException e) {
    throw new RuntimeException(e,"Error reading promoter site",26,3);
}
}
}

```

```

////////////////////////////////////
// ReadStream.java
//
// Next available error code: 28-11
////////////////////////////////////

import java.io.*;

class ReadStream
{
    private final static String eofMarker = "[EOF]";
    private BufferedReader in;

    ReadStream (FileInputStream s)
    {
        in = new BufferedReader(new InputStreamReader(s));
    }

    ReadStream (DataInputStream s)
    {
        in = new BufferedReader(new InputStreamReader(s));
    }

    /* DELETED THIS METHOD UPON CONVERTING TO BufferedReader CLASS
    DataInputStream getInputStream ()
    {
        return in;
    }
    */

    void close () throws Exception
    {
        try {
            in.close();
        }
        catch (IOException e) {
            throw new Exception("Error while closing connection",28,7);
        }
    }
}

////////////////////////////////////

```

```

// myReadLine ()
//
// Reads a line from the BufferedReader
//////////
private String myReadLine () throws Exception
{
    String buf;

    try {
        buf = in.readLine();
    }
    catch (IOException e) {
        throw new Exception(e,"Error reading stream",28,6);
    }

    return buf;
}

String readLine () throws Exception
{
    String buf;

    buf = myReadLine();

    // If we have a non-null string then check it for errors
    // and the special [EOF] message
    if (buf != null) {

        // See if we have received an error
        if (buf.startsWith("[Error: ") {
            String buf2;

            // Read in the error reference and message
            if ((buf2 = myReadLine()) == null)
                throw new Exception("Unexpected EOF",28,10);
            if (!(buf2.startsWith("[Error: "))) {
                throw new Exception(
                    "Error reading \"[Error : \" received \"+
                    buf2,28,9);
            }
            readMessage("[Error]");

            // Extract the error message and reference number
            buf = buf.substring(8,buf.length()-1);
            buf2 = buf2.substring(8,buf2.length()-1);

```

```

        // Throw the error exception
        throw new ErrorException(buf,"Ref: "+buf2);
    }
    else {
        // Check for the special [EOF] signal
        buf = (isEOF(buf) ? null : buf);
    }
}

```

```

// Return the string
return buf;
}

```

```

private boolean isEOF (String buf)
{
    return buf.equals(eofMarker);
}

```

```

////////////////////////////////////
// readFirstNonBlankLine ()
//
// Reads and returns the next non-blank line
////////////////////////////////////
String readFirstNonBlankLine () throws ErrorException
{
    String buf;

    while ((buf = readLine()) != null)
        if (buf.length() > 0) break;
    if (buf == null)
        throw new ErrorException("Unexpected EOF",28,8);

    return buf;
}

```

```

////////////////////////////////////
// readPrefix (prefix)
//
// Reads in the next non-blank line which is assumed to start with
// the specified prefix. If it does it removes the prefix and
// returns the remainder of the line. If the next non-blank line
// does not start with the specified prefix an Exception is thrown
////////////////////////////////////

```

```
String readPrefix (String prefix) throws RecognitionException
{
    String buf = readFirstNonBlankLine();

    if (buf == null) throw new RecognitionException(
        "Unexpected end of file reading \"" + prefix + "\",28,4);

    if (buf.startsWith(prefix))
        buf = buf.substring(prefix.length());

    else throw new RecognitionException(
        "Error reading \"" + prefix + "\" received \"" + buf + "\",28,2);

    return buf;
}
```

```
////////////////////////////////////
// readMessage (message)
//
// Reads the next non-blank line. If that line is not equal to
// the specified message an Exception is thrown.
////////////////////////////////////
void readMessage (String message) throws RecognitionException
{
    String buf = readFirstNonBlankLine();

    if (buf == null) throw new RecognitionException(
        "Unexpected end of file reading \"" + message + "\",28,5);

    if (!buf.equals(message)) throw new RecognitionException(
        "Error Reading \"" + message + "\" received \"" + buf + "\",28,3);
}
```

```
String readBlock (int maxLen, String end) throws RecognitionException
{
    int i, j, len = end.length();
    char[] buf = new char[maxLen];

    for (i=0; i<maxLen; i++) {

        // Read a character
        try {
            buf[i] = (char)(in.read());
        }
    }
}
```



```

        catch (IOException e) {
            throw new RuntimeException(e,"Error reading block",28,11);
        }

// See if we have the end sequence
    if ((len > 0) && (i >= len-1)) {
        for (j=0; j<len; j++) {
            if (buf[i-len+1+j] != end.charAt(j)) break;
        }
        if (j == len) {
            i++;
            break;
        }
    }

    return new String(buf,0,i);
}
}

```

```
////////////////////////////////////
```

```
// Rpt.java
```

```
//
```

```
// Next available error code: 28-4
```

```
////////////////////////////////////
```

```
import java.io.IOException;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;
```

```
class Rpt extends GrailItem
```

```
{
    String type;
    float score;
```

```
    static String buttonLabel, identifier, requestMessage, fieldNames;
```

```
    Rpt ()
```

```
    {
        solid = true;
        color = Grail.rptColor;
        highlightColor = Grail.rptHighlightColor;
        buttonLabel = "Reptv DNA";
        identifier = "rptv";
        requestMessage = "dr.";
        fieldNames = " Type Str Start End Score";
    }
```

```
    public String toString ()
```

```
    {
        StringBuffer s = new StringBuffer(100);
        int i; for (i=0; i<50; i++) s.append(' ');

        try
        {
            s.insert(0,type);
            s.insert(8,dirn);
            s.insert(11,start);
            s.insert(20,end);
            s.insert(29,score);
        }
    }
```

```

    catch (ArrayIndexOutOfBoundsException e) {}
    catch (StringIndexOutOfBoundsException e) {}

    return s.toString().trim();
}

GrailItem newItem () { return new Rpt(); }

String getLabel () { return buttonLabel; }

String getIdentifier () { return identifier; }

String getRequestMessage () { return requestMessage; }

String getFieldNames () { return fieldNames; }

void write (WriteStream out) throws ErrorException
{
    if ((!Grail.fileCompMode) || (dirn == 'f')) {
        out.writeBytes (str2str(type,8) + " " + dirn + " " +
                        int2str(start,6) + " " +
                        int2str(end,6) + " " + float2str(score,6,2) + "\n");
    }
    else {
        out.writeBytes (str2str(type,8) + " " + dirn + " " +
                        int2str(1+Grail.sequenceLength-end,6) + " " +
                        int2str(1+Grail.sequenceLength-start,6) + " " +
                        float2str(score,6,2) + "\n");
    }
}

void read (ReadStream in) throws ErrorException
{
    String buf;
    StringTokenizer st;

    // Read the first non-blank line
    st = new StringTokenizer(in.readFirstNonBlankLine());

    try {

        // Grab the type name
        type = st.nextToken();

        // Grab the direction character
        if ((buf = st.nextToken()).length() != 1)

```

```

        throw new RuntimeException("Error reading rpt DNA site",27,1);
        dirn = buf.charAt(0);

// Parse out the start position
        start = Integer.parseInt(st.nextToken());

// Parse out the end position
        end = Integer.parseInt(st.nextToken());

// Grab the score
        score = Float.valueOf(st.nextToken()).floatValue();

        //      Correct for complementary mode
        if ((Grail.compMode) && (dirn == 'r')) {
            int tmp;
            tmp = 1 + Grail.sequenceLength - end;
            end = 1 + Grail.sequenceLength - start;
            start = tmp;
        }
    }

    catch (NumberFormatException e) {
        throw new RuntimeException(e,"Error reading rpt DNA site",27,2);
    }

    catch (NoSuchElementException e) {
        throw new RuntimeException(e,"Error reading rpt DNA site",27,3);
    }
}
}
}

```

```

////////////////////
// SaveChoice.java
//
// Next available error code: 29-1
////////////////////

import java.io.*;
import java.awt.*;
import java.net.*;
import java.util.Vector;
import java.applet.Applet;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;

class SaveChoice extends ThreadedFrame
{
    final static int CANCEL = 0;
    final static int SAVE = 1;

    int mode;
    Checkbox cb;
    TextField selectionBox;
    Button saveButton, cancelButton;
    Thread callingThread;

    SaveChoice (String filename)
    {
        super("Save");

        mode = SAVE;

        // Make the panel that holds the selection box
        Panel selectionPanel = new Panel();
        // Replace the panel's default FlowLayout with a BorderLayout
        selectionPanel.setLayout(new BorderLayout());
        // Add the "Selection" label which, by default, aligns to the left
        selectionPanel.add("North",new Label("Selection"));
        // Add the selection TextField
        selectionPanel.add("South",selectionBox = new TextField(filename));
        selectionBox.setEditable(true);

        // Make the panel that holds the checkbox
        Panel checkboxPanel = new Panel();
        checkboxPanel.add(cb = new Checkbox("Complementary Mode"));
    }
}

```

```

        cb.setState(Grail.fileCompMode);

// Make the panel that holds the buttons
    Panel buttonPanel = new Panel();
// Replace default FlowLayout with a 1-row, 2-column grid
    buttonPanel.setLayout(new GridLayout(1,2));
// Place "Save" button in its own panel and add to the button panel
    Panel tempPanel = new Panel();
    tempPanel.add(saveButton = new Button("Save"));
    buttonPanel.add(tempPanel);
// Place "Cancel" button in its own panel and add to the button panel
    tempPanel = new Panel();
    tempPanel.add(cancelButton = new Button("Cancel"));
    buttonPanel.add(tempPanel);

// Put it all together in a 3-row, 1-column grid
    setLayout(new GridLayout(3,1));
    add(selectionPanel);
    add(checkboxPanel);
    add(buttonPanel);

// Finally, use pack to resize the window
    pack();
}

public boolean myAction (Event evt, Object arg) throws Exception
{
// Deal with the cancel button
    if (evt.target == cancelButton) {

        // Hide the window - the main frame will dispose of it
        hide(CANCEL);
        return true;
    }

// Deal with the save button
    else if (evt.target == saveButton) {

        // Hide the window - the main frame will dispose of it
        Grail.fileCompMode = cb.getState();
        hide(mode);
        return true;
    }

// Deal with the Selection TextField

```

```

        else if (evt.target == selectionBox) {

            // Hide window and return
            hide(mode);
            return true;
        }

        return false;
    }

    public void setVisible (boolean visvar)
    {
        if (visvar)
        {
            // Show the window
            super.show();

            // Find out what thread I'm in
            callingThread = Thread.currentThread();

            // The fileChoice show() method blocks until the user selects a file
            // or cancels. I originally implemented the blocking by calling
            // suspend on the thread here, and calling resume on the thread
            // in the action() method when they hit the OK button. However
            // under netscape, calls to suspend() and resume() generate security
            // exceptions. For this reason I just go to sleep here until
            // the action() method sets the callingThread pointer to null
            // (when the user hits the OK button.) I poll the callingThread
            // pointer once every second and drop out of the loop as soon as
            // I detect that it is null.
            while (callingThread != null) {
                try { callingThread.sleep(1000); }
                catch (InterruptedException e) {}
            }
            else
            {
                super.setVisible(false);
            }
        }
    }

    public void hide (int md)
    {
        mode = md;
    }

```

```
setVisible(false);
```

```
// Tell the calling thread to resume. I originally called  
// suspend() on the calling thread and resumed it here. However  
// this generated a security exception under netscape so  
// I now just have the thread sleep until it detects that  
// the currentThread pointer has been set to null. Here I  
// set currentThread = null which will cause the show()  
// method to terminate and return.  
    callingThread = null;  
}
```

```
String getSelection ()  
{  
    return ((mode == CANCEL) ? null : selectionBox.getText());  
}  
}
```



```

//////////
// SequenceCanvas.java
//////////

import java.awt.*;
import java.util.Vector;
import java.awt.AWTEvent;

class SequenceCanvas extends Canvas
{
    int charWidth, charHeight, numChars, seqLengthMinus1;
    int fmin, fmax, rmin, rmax, mid;
    int topStringY, botStringY, topTranslationY, botTranslationY;
    int topExonY, botExonY, topModelY, botModelY;
    int topShadowY, botShadowY, topRulerY, botRulerY, topPolyaY, botPolyaY;
    int exonBarHeight, tickLength, cpgY, topRptY, botRptY, topPromY, botPromY;
    int rptBarHeight, cpgBarHeight, promBarHeight, polyaBarHeight;
    // MyScrollbar scrollbar;
    Scrollbar scrollbar;
    MyLabel forwardLeft, forwardRight, revLeft, revRight;
    FontMetrics fontMetrics;
    Grail myParent;

    // Graphics buffer and size stuff
    Font boldEvenlySpacedFont, plainEvenlySpacedFont;
    int bufferWidth, bufferHeight;
    Dimension prefSize, minSize;
    Rectangle clipRect;
    Image bufferImage;
    Graphics buffer;

    final Color GColor = Color.green;
    final Color CColor = Color.cyan;
    final Color AColor = Color.lightGray;
    final Color TColor = Color.pink;
    final static Color exonBarColor = Color.cyan;
    final Color exonHighlightColor = Color.yellow;
    final Color modelExonBarColor = Color.yellow;
    final Color modelHighlightColor = Color.cyan;
    final static Color shadowBarColor = Color.lightGray;
    final Color shadowHighlightColor = Color.yellow;
    final Color exonTranslationColor = Color.cyan;
    final Color shadowTranslationColor = Color.lightGray;
    final Color modelTranslationColor = Color.yellow;

```

```

final Color rulerColor = Color.red;

SequenceCanvas (Grail g)
{
    // Save pointer to my parent
    myParent = g;

    // Set the background color
    setBackground(Color.black);

    // Set the minimum and preferred dimensions. The window takes up
    // 128 pixels plus the width of this canvas.
    int screenWidth = Toolkit.getDefaultToolkit().getScreenSize().width;
    prefSize = new Dimension(screenWidth-160,212);
    minSize = new Dimension(screenWidth-160,212);

    // Set up the position labels
    forwardLeft = new MyLabel(6,Label.LEFT);
    forwardRight = new MyLabel(6,Label.RIGHT);
    revLeft = new MyLabel(6,Label.LEFT);
    revRight = new MyLabel(6,Label.RIGHT);

    // Set up the scroll bar
    scrollbar = new Scrollbar(Scrollbar.HORIZONTAL);
}

public Dimension getPreferredSize ()
{
    return prefSize;
}

public Dimension getMinimumSize ()
{
    return minSize;
}

public void paint (Graphics g)
{
    update(g);
}

void updateLabels ()
{
    forwardLeft.setText(Integer.toString((fmin=scrollbar.getValue()+1));
    forwardRight.setText(Integer.toString((fmax=fmin+numChars-1)+1));
    revLeft.setText(Integer.toString((rmax=seqLengthMinus1-fmin)+1));
}

```

```

    revRight.setText(Integer.toString((rmin=seqLengthMinus1-fmax)+1));
}

void setSequence ()
{
    seqLengthMinus1 = Grail.sequence.length()-1;
}

void fixSize ()
{
    int ascent, descent, leading, barSep;

// Get the fonts
    boldEvenlySpacedFont = new Font("Courier",Font.BOLD,14);
    plainEvenlySpacedFont = new Font("Courier",Font.PLAIN,14);

// Set up the strings
    setSequence();

// Get the dimensions of a single character
    fontMetrics = getFontMetrics(plainEvenlySpacedFont);
    charHeight = fontMetrics.getHeight();
    ascent = fontMetrics.getAscent();
    descent = fontMetrics.getDescent();
    leading = fontMetrics.getLeading();

    charWidth = fontMetrics.charWidth('A');

    barSep = 3;
    tickLength = 3;
    exonBarHeight = 5;
    rptBarHeight = 5;
    promBarHeight = 5;
    polyabarHeight = 5;
    cpgBarHeight = 5;

    cpgY = cpgBarHeight/2;
    botRptY = cpgY + barSep + 1;
    topRptY = botRptY + rptBarHeight - 1;
    botPromY = topRptY + barSep + 1;
    topPromY = botPromY + promBarHeight - 1;
    botPolyaY = topPromY + barSep + 1;
    topPolyaY = botPolyaY + polyabarHeight - 1;
    topTranslationY = topPolyaY + barSep + descent + 1;
    botTranslationY = topPolyaY + barSep + ascent;
    topRulerY = botRulerY = topTranslationY + ascent + barSep;

```

```

    topStringY = topRulerY + tickLength + descent + 2;
    botStringY = topRulerY + tickLength + ascent + 1;
    botExonY = topStringY + ascent + barSep + 1;
    topExonY = botExonY + exonBarHeight - 1;
    botShadowY = topExonY + barSep + 1;
    topShadowY = botShadowY + exonBarHeight - 1;
    botModelY = topShadowY + barSep + 1;
    topModelY = botModelY + exonBarHeight - 1;
}

void jumpToBase (int base)
{
    scrollbar.setValue(base - numChars/2);
    repaint();
}

void drawRulers (Graphics g)
{
    int i, lo = charWidth, hi = lo+numChars*charWidth;

    // Draw the horizontal lines
    g.setColor(rulerColor);
    g.drawLine(lo,mid-topRulerY,hi,mid-topRulerY);
    g.drawLine(lo,mid+botRulerY,hi,mid+botRulerY);

    // Put the tick marks on the top line
    for (i=(1+(10-(fmin%10))%10)*charWidth; i<=hi-charWidth;
        i+=10*charWidth)
        g.drawLine(i,mid-topRulerY,i,mid-topRulerY-tickLength);

    // Put the tick marks on the bot line
    for (i=hi-((rmax+1)%10)*charWidth; i>=lo+charWidth; i-=10*charWidth)
        g.drawLine(i,mid+botRulerY,i,mid+botRulerY+tickLength);
}

public void update (Graphics g)
{
    try {
        Rectangle r = new Rectangle();

        // If the buffer does not exist, or is a different size than
        // the current canvas, make a new one.
        if ((buffer == null) || (bufferWidth != getSize().width) ||
            (bufferHeight != getSize().height)) {
            bufferWidth = getSize().width;
            bufferHeight = getSize().height;
        }
    }
}

```

```

        bufferImage = createImage(bufferWidth,bufferHeight);
        buffer = bufferImage.getGraphics();

    // Get the clipRect - images don't have one it appears
    clipRect = new Rectangle(0,0,bufferWidth,bufferHeight);
    }

    // Blank out the buffer
    buffer.setColor(getBackground());
    buffer.fillRect(0,0,bufferWidth,bufferHeight);
    buffer.setColor(getForeground());

    // Compute the number of characters that will fit in the window
    numChars = (bufferWidth/charWidth)-2;
    if (numChars > seqLengthMinus1+1) numChars = seqLengthMinus1+1;

    // Set the font
    buffer.setFont(plainEvenlySpacedFont);

    // Update the position labels
    updateLabels();

    // Set the scrollbar values appropriately
    scrollbar.setValues(fmin,numChars,0,1+seqLengthMinus1-numChars);

    // Get the midpoint of the window
    mid = bufferHeight/2;

    // Draw the Rpt sites
    if (Grail.rpt.isShowing()) {
        r.setBounds(mid+botRptY,mid-topRptY,fmin,rptBarHeight);
        Grail.rpt.draw(r,1/(float)charWidth,buffer,clipRect);
    }

    // Draw the CpG Islands
    if (Grail.cpg.isShowing()) {
        r.setBounds(mid-cpgY,mid-cpgY,fmin,cpgBarHeight);
        Grail.cpg.draw(r,1/(float)charWidth,buffer,clipRect);
    }

    // Draw the Pol II Promoter sites
    if (Grail.promoter.isShowing()) {
        r.setBounds(mid+botPromY,mid-topPromY,fmin,promBarHeight);
        Grail.promoter.draw(r,1/(float)charWidth,buffer,clipRect);
    }

```

```

// Draw the Polya sites
if (Grail.polya.isShowing()) {
    r.setBounds(mid+botPolyaY,mid-topPolyaY,fmin,polyaBarHeight);
    Grail.polya.draw(r,1/(float)charWidth,buffer,clipRect);
}

// Put the bars over the exons
Rectangle b = new Rectangle
    (mid+botExonY,mid-topExonY,fmin,exonBarHeight);
Grail.exons[Grail.ver].draw(b,mid-topShadowY,mid+botShadowY,
    mid-topTranslationY,mid+botTranslationY,1/(float)charWidth,
    buffer,clipRect);
if (Grail.ver == 2) {
    b.setBounds(mid+botModelY,mid-topModelY,fmin,exonBarHeight);
    Grail.geneModels.draw(b,mid-topTranslationY,
        mid+botTranslationY,1/(float)charWidth,buffer,clipRect);
}

// Show the rulers
drawRulers(buffer);

// Draw the sequence (G and C in bold, A and T in plain font.)
int i,j=fmin,pos=charWidth;
for (i=0,j=fmin; i<numChars; i++) {
    switch (Grail.sequence.charAt(j)) {
        case 'A': buffer.setFont(plainEvenlySpacedFont);
            buffer.setColor(AColor);
            buffer.drawString("A",pos,mid-topStringY);
            buffer.setColor(TColor);
            buffer.drawString("T",pos,mid+botStringY);
            break;
        case 'C': buffer.setFont(boldEvenlySpacedFont);
            buffer.setColor(CColor);
            buffer.drawString("C",pos,mid-topStringY);
            buffer.setColor(GColor);
            buffer.drawString("G",pos,mid+botStringY);
            break;
        case 'T': buffer.setFont(plainEvenlySpacedFont);
            buffer.setColor(TColor);
            buffer.drawString("T",pos,mid-topStringY);
            buffer.setColor(AColor);
            buffer.drawString("A",pos,mid+botStringY);
            break;
        case 'G': buffer.setFont(boldEvenlySpacedFont);
            buffer.setColor(GColor);
            buffer.drawString("G",pos,mid-topStringY);
    }
}

```

```

        buffer.setColor(CColor);
        buffer.drawString("C",pos,mid+botStringY);
        break;
    default: break;
}
j++; pos+=charWidth;
}

// And paste the buffer onto the screen
g.drawImage(bufferImage,0,0,this);
}

// Pass any errors to the main window
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}
}

public boolean handleEvent (Event evt)
{
    try {
        if (evt.id == Event.MOUSE_DOWN)
        {
            // Convert the X position to a forward strand base number
            int base = (evt.x/charWidth)+fmin;

            // Convert the Y position to a from-midpoint distance
            int y = Math.abs(evt.y-mid);
            char[] strand = {'f','s'};
            if (evt.y > mid) strand[0] = 'r';

            // Hit the Polya sites, if appropriate
            Polya.clickWidth = 0;
            if (Grail.polya.isShowing() &&
                (y >= botPolyaY) && (y <= topPolyaY))
                if (Grail.polya.mouseClick(base,strand))
                {
                    if (Grail.polya.itemList.getSelectedIndex() != -1)
                        Grail.mainCanvas.jumpToBase(base);
                    else Grail.mainCanvas.repaint();
                    repaint();
                    return true;
                }
        }

        // Hit the Pol II promoters, if appropriate
        if (Grail.promoter.isShowing() &&

```

```

(y >= botPromY) && (y <= topPromY))
if (Grail.promoter.mouseClick(base,strand))
{
    if (Grail.promoter.itemList.getSelectedIndex() != -1)
        Grail.mainCanvas.jumpToBase(base);
    else Grail.mainCanvas.repaint();
    repaint();
    return true;
}

// Hit the Rpt's, if appropriate
if (Grail.rpt.isShowing() && (y >= botRptY) && (y <= topRptY))
if (Grail.rpt.mouseClick(base,strand))
{
    if (Grail.rpt.itemList.getSelectedIndex() != -1)
        Grail.mainCanvas.jumpToBase(base);
    else Grail.mainCanvas.repaint();
    repaint();
    return true;
}

// Hit the standard Exons, if appropriate
if ((y >= botExonY) && (y <= topExonY))
if (Grail.exons[Grail.ver].mouseClick(base,strand))
{
    if (Grail.exons[Grail.ver].itemList.
        getSelectedIndex() != -1)
        Grail.mainCanvas.jumpToBase(base);
    else Grail.mainCanvas.repaint();
    repaint();
    return true;
}

// Hit the alternate Exons, if appropriate
strand[1] = 'a';
if (Grail.exons[Grail.ver].showingShadows &&
    (y >= botShadowY) && (y <= topShadowY))
if (Grail.exons[Grail.ver].mouseClick(base,strand))
{
    if (Grail.exons[Grail.ver].itemList.
        getSelectedIndex() != -1)
        Grail.mainCanvas.jumpToBase(base);
    else Grail.mainCanvas.repaint();
    repaint();
    return true;
}

```



```

// Hit the model exons if appropriate
if (Grail.assembleMenu.isEnabled() &&
    (y >= botModelY) && (y <= topModelY))
    if (Grail.geneModels.mouseClick(base,strand))
    {
        int m;
        if ((m=Grail.geneModels.itemList.
            getSelectedIndex()) != -1)
            if (((Gene_Model)Grail.geneModels.item.
                elementAt(m)).
                exons.itemList.getSelectedIndex() != -1)
                Grail.mainCanvas.jumpToBase(base);
            else Grail.mainCanvas.repaint();
        repaint();
        return true;
    }

// Hit the CpG islands, if appropriate
strand[0] = 'f';
if (Grail.cpg.isShowing() && (y <= cpgY))
    if (Grail.cpg.mouseClick(base,strand))
    {
        if (Grail.cpg.itemList.getSelectedIndex() != -1)
            Grail.mainCanvas.jumpToBase(base);
        else Grail.mainCanvas.repaint();
        repaint();
        return true;
    }
}

return false;
}

// Pass any errors up to the main window
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

return false;
}
}

```

```
////////////////////////////////////
```

```
// SequenceWindow.java
```

```
//
```

```
// Next available error code: 22-2
```

```
////////////////////////////////////
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
class SequenceWindow extends ThreadedFrame  
    implements AdjustmentListener
```

```
{
```

```
    SequenceCanvas sequenceCanvas;
```

```
    MenuItem printButton, closeButton;
```

```
    Grail myParent;
```

```
    final static String title = "DNA SEQUENCE";
```

```
    final static String buttonName = "DNA Sequence";
```

```
    SequenceWindow (Grail g)
```

```
    {
```

```
        // Set window title, layout manager and size
```

```
        super (title + ": Sequence: None");
```

```
        setLayout(new BorderLayout());
```

```
        // Store pointer to grail
```

```
        myParent = g;
```

```
        // Set up the menus
```

```
        printButton = new MenuItem("Print"); printButton.setEnabled(false);
```

```
        closeButton = new MenuItem("Close");
```

```
        Menu fileMenu = new Menu("File");
```

```
        fileMenu.add(printButton);
```

```
        fileMenu.addSeparator();
```

```
        fileMenu.add(closeButton);
```

```
        MenuBar mb = new MenuBar();
```

```
        mb.add(fileMenu);
```

```
        setMenuBar(mb);
```

```
        // Make the main drawing canvas
```

```
        sequenceCanvas = new SequenceCanvas (myParent);
```

```
        // Make the panel that will hold the left position labels
```

```
        Panel leftLabelPanel = new Panel();
```

```

leftLabelPanel.setLayout(new BorderLayout());
leftLabelPanel.setBackground(Color.black);
leftLabelPanel.setForeground(Color.white);
leftLabelPanel.setFont(new Font("Courier",Font.BOLD,14));
leftLabelPanel.add("North",sequenceCanvas.forwardLeft);
leftLabelPanel.add("South",sequenceCanvas.revLeft);

// Make the panel that will hold the right position labels
Panel rightLabelPanel = new Panel();
rightLabelPanel.setLayout(new BorderLayout());
rightLabelPanel.setBackground(Color.black);
rightLabelPanel.setForeground(Color.white);
rightLabelPanel.setFont(new Font("Courier",Font.BOLD,14));
rightLabelPanel.add("North",sequenceCanvas.forwardRight);
rightLabelPanel.add("South",sequenceCanvas.revRight);

// Add a dummy label to the label panels to give them constant width
leftLabelPanel.add("Center",new Label("  "));
rightLabelPanel.add("Center",new Label("  "));

// Add all the items to the window
add("Center",sequenceCanvas);
add("West",leftLabelPanel);
add("East",rightLabelPanel);
add("South",sequenceCanvas.scrollbar);

// Call pack() to size the thing
pack();
}

void reshaw ()
{
    setTitle(title + ":  " + Grail.seqname + " Length: " +
        Grail.sequence.length());
    pack();
    show();
    sequenceCanvas.fixSize();
}

public boolean myHandleEvent (Event evt) throws ErrorException
{
    try {
        switch (evt.id)
        {

            case AdjustmentEvent.UNIT_INCREMENT:

```

```

        case AdjustmentEvent.UNIT_DECREMENT:
        case AdjustmentEvent.BLOCK_INCREMENT:
        case AdjustmentEvent.BLOCK_DECREMENT:
        case AdjustmentEvent.TRACK:

            sequenceCanvas.repaint();
            return true;
        case Event.WINDOW_DESTROY:
            setVisible(false);
            return true;
        default: break;
    }

    return false;
}

// Pass any errors to Grail window
catch (Exception e) {
    myParent.dispatchEvent(new ExceptionEvent(myParent,e));
}

return false;
}

public boolean myAction (Event evt, Object arg)
{
    String label = (String)arg;

    if (evt.target instanceof MenuItem)
    {
        if (label.equals(closeButton.getLabel())) setVisible(false);
        return true;
    }

    return false;
}
}

```

```

//////////
// StringOp.java
//////////

class StringOp
{
    static String reverse (String s)
    {
        int i, j=0;
        char[] c = new char[s.length()];

        for (i=s.length()-1; i>=0; i--) c[j++] = s.charAt(i);

        return new String(c);
    }

    static String reverseSequence (String s)
    {
        int i, j=0;
        char[] c = new char[s.length()];

        for (i=s.length()-1; i>=0; i--) {
            switch (s.charAt(i)) {
                case 'A': c[j++] = 'T'; break;
                case 'C': c[j++] = 'G'; break;
                case 'T': c[j++] = 'A'; break;
                case 'G': c[j++] = 'C'; break;
            }
        }

        return new String(c);
    }
}

```

```

//////////
// ThreadedFrame.java
//////////

import java.awt.*;
import java.awt.event.*;

class ThreadedFrame extends Frame
    implements Runnable,
        ItemListener,
        ActionListener,
        WindowListener,
        AdjustmentListener
{
    Object eventArg;
    AWTEvent currentEvent;
    Thread eventThread;
    boolean[] menuStatus;
    boolean[] componentStatus;
    boolean windowStatus = true;

    ThreadedFrame (String s)
    {
        super(s);
    }

    // Set up JDK 1.1.3 Event Methods

    public void itemStateChanged ( ItemEvent evt )
    {
        if (evt.getSource() instanceof List)

            J102_handleEvent (evt);

        else

            J102_action (evt);
    }

    public void actionPerformed ( ActionEvent evt )
    {

```

```

        J102_action (evt);
    }

    public void windowClosing ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void windowOpened ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void windowIconified ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void windowDeiconified ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void windowClosed ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void windowActivated ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void windowDeactivated ( WindowEvent evt)
    {
        J102_handleEvent (evt);
    }

    public void adjustmentValueChanged (AdjustmentEvent evt)
    {
        J102_handleEvent (evt);
    }

```

////////////////////////////////////

```

// J102_action (evt)
//
// Handles all action events by either firing up a thread to
// deal with them (if actionNeedsAThread() returns true),
// having the descendant myAction handle them, or passing
// the event on up the component heirarchy. If a thread is
// needed to process an action then the thread is spawned and
// calls the descendant threadedAction() method. If an exception
// occurs at any point during the handling of the action a
// thread is fired up which calls handleException().
////////////////////////////////////
public boolean J102_action (AWTEvent evt)
{
    boolean needAThread;

    // Bail immedialtely if the window is disabled
    if (!isEnabled()) return false;

    // See if we need a thread to handle this action event
    try {
        needAThread = actionNeedsAThread(evt);
    }

    // If the call to actionNeedsAThread generated an exception
    // then fire up a thread that will call handleException
    catch (Exception e) {
        startEventThread("error",null,e);
        return true;
    }

    // If this event needs to be handled in a thread then
    // fire up a thread to deal with it, and return true
    if (needAThread) {
        startEventThread("action",evt,null);
        return true;
    }

    // If this action does not need to be handled in a thread
    // then see if it can be handled directly
    else {
        try {
            if (myAction(evt) ) return true;
        }

        // If the call to myAction() generates an exception then
        // start up a thread that will call handleException()

```



```

        catch (Exception e) {
            startEventThread("error",null,e);
            return true;
        }
    }

    // If we didn't handle the event then return false
    return false;
}

////////////////////////////////////
// J102_handleEvent (evt)
//
// Handles all general events by either firing up a thread to
// deal with them (if eventNeedsAThread() returns true),
// having the descendant myHandleEvent handle them, or passing
// the event on up the component heirarchy. If a thread is
// needed to process an event then the thread is spawned and
// calls the descendant threadedHandleEvent() method. If any
// of the event-handling code generates an exception a thread
// is fired up which calls handleException().
////////////////////////////////////
public boolean J102_handleEvent (AWTEvent evt)
{
    boolean needAThread;

    // Bail immediately if the window is disabled
    if (!isEnabled()) return super.processEvent(evt);

    try {
        needAThread = eventNeedsAThread(evt);
    }

    // If the call to eventNeedsAThread generated an exception
    // then fire up a thread that will call handleException
    catch (Exception e) {
        startEventThread("error",null,e);
        return true;
    }

    // If the event needs a thread to deal with it then
    // fire one up
    if (needAThread) {

```

```

        startEventThread("handleEvent",evt,null);
        return true;
    }

    // If the event doesn't need a thread then see if it can
    // be handled directly.
    else {
        try {
            if (myHandleEvent(evt)) return true;
        }

        // If the call to myHandleEvent() generates an exception then
        // start up a thread that will call handleException()
        catch (Exception e) {
            startEventThread("error",null,e);
            return true;
        }
    }

    // They didn't handle it so pass it on up the component heirarchy
    return super.processEvent(evt);
}

```

```

////////////////////////////////////
// startEventThread (caller, evt, arg)
//
// This starts a thread that will respond to user actions and
// events. The "caller" variable is a string that identifies
// which method invoked the event-handling thread.
////////////////////////////////////
void startEventThread (String caller, AWTEvent evt, Object arg)
{
    // Store the event and argument in instance variables so
    // that they are accessible to the thread.
    currentEvent = evt;
    eventArg = arg;

    // Allow the user to do some stuff before the thread starts
    preThreadInit();

    // And go ahead and spawn the new thread.
    eventThread = new Thread(this,caller);

```

```

        eventThread.start();
    }

////////////////////////////////////
// run()
//
// This is the entry point for the event-handling thread.
// All error messages bubble up to this method and are
// caught here and handled by throwing up the error
// window containing the detail message and reference no.
////////////////////////////////////
    public void run ()
    {
        try {

            // RESPOND TO AN ERROR EXCEPTION FROM ACTION OR
HANDLEEVENT
            if (eventThread.getName().equals("error")) {
                throw((Exception)eventArg);
            }

            // RESPOND TO A CALL TO THE ACTION METHOD
            else if (eventThread.getName().equals("action")) {
                threadedAction(currentEvent);
            }

            // RESPOND TO A CALL TO THE HANDLEEVENT METHOD
            else if (eventThread.getName().equals("handleEvent")) {
                threadedHandleEvent(currentEvent);
            }
        }

        // CATCH ANY ERRORS
        catch (Exception e) {
            handleException(e);
        }

        // TIDY UP BEFORE RETURNING
        postThreadCleanUp();
    }

    void disableMenu (int i)
    {

```

```

MenuBar mb = getMenuBar();

if (mb != null) {
    mb.getMenu(i).setEnabled(false);
    menuStatus[i] = false;
}
}

void disableMenu (String label)
{
    int i, n;
    MenuBar mb = getMenuBar();

    if (mb != null) {
        n = mb.getMenuCount();
        for (i=0; i<n; i++) {
            if (mb.getMenu(i).getLabel().equals(label)) {
                disableMenu(i);
                break;
            }
        }
    }
}

```

```

void enableMenu (int i)
{
    MenuBar mb = getMenuBar();

    if (mb != null) {
        mb.getMenu(i).setEnabled(true);
        menuStatus[i] = true;
    }
}

```

```

void enableMenu (String label)
{
    int i, n;
    MenuBar mb = getMenuBar();

    if (mb != null) {
        n = mb.getMenuCount();
        for (i=0; i<n; i++) {
            if (mb.getMenu(i).getLabel().equals(label)) {

```

```

        enableMenu(i);
        break;
    }
}
}
}

```

```

public void setEnabled (boolean cond)
{
    if (cond) {
        //////////////////////////////////////
        // enable()
        //
        // Enables this frame and all menus that were enabled
        // prior to the call to disable(). Note that any menus
        // enabled in the interim period between the last call
        // to disable and the current call to enable will be
        // left enabled.
        //////////////////////////////////////
        if (!isEnabled()) {
            int i;
            MenuBar mb = getMenuBar();

            // Disable all of the components in the frame. Originally I
            // just called disable() on the frame here - however on PCs
            // (bug city) this prevented the display of the WAIT_CURSOR.
            // So now i individually disable each subcomponent of the
            // frame.

            // Find out how many components there are
            i = getComponentCount();

            // If componentStatus matches it (i.e. we have previously
            // called disable) then restore each component
            // to its previous status
            if ((componentStatus != null ) && (componentStatus.length == i)) {
                while (i-- > 0) {
                    if (componentStatus[i]) getComponent(i).setEnabled(true);
                }
            }

            // If the size of componentStatus does not correspond to
            // the current number of components, then just enable them all
            else {
                while (i-- > 0) getComponent(i).setEnabled(true);
            }
        }
    }
}

```

```

    }

// If we have a menubar then enable it
if (mb != null) {

    // Find out how many menus there are in the menu bar
    i = mb.getMenuCount();

    // If menuStatus matches it (i.e. we have previously
    // called disable) then restore each menu in the menu
    // bar to its previous status
    if ((menuStatus != null) && (menuStatus.length == i)) {
        while (i-- > 0) {
            if (menuStatus[i]) mb.getMenu(i).setEnabled(true);
        }
    }

    // If the size of menuStatus does not correspond to
    // the current number of menus, then just enable them all
    else {
        while (i-- > 0) mb.getMenu(i).setEnabled(true);
    }
}

// Switch to the default cursor
setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
}

// Store status
windowStatus = true;
}

////////////////////
// disable()
//
// Disables this frame and all its menus
////////////////////
else {
    // Disable this window (if appropriate)
    if (isEnabled()) {
        int i;
        MenuBar mb = getMenuBar();

        // Switch to the wait cursor
        setCursor(new Cursor(Cursor.WAIT_CURSOR));
    }
}

```

```

// If we have a menubar then go ahead and disable it
if (mb != null) {
    // Find out how many menus there are in the menu bar
    i = mb.getMenuCount();

    // Make sure menuStatus is the appropriate size
    if ((menuStatus == null) || (menuStatus.length != i))
        menuStatus = new boolean[i];

    // Loop through each of the menus and for each one ...
    while (i-- > 0) {
        // Store its current status
        menuStatus[i] = mb.getMenu(i).isEnabled();

        // And disable it
        mb.getMenu(i).setEnabled(false);
    }
}

// Find out how many components there are
i = getComponentCount();

// Make sure componentStatus is the appropriate size
if ((componentStatus == null) || (componentStatus.length != i))
    componentStatus = new boolean[i];

// Loop through each of the components and for each one ...
while (i-- > 0) {
    // Store its current status
    componentStatus[i] = getComponent(i).isEnabled();

    // And disable it
    getComponent(i).setEnabled(false);
}

// Store status
windowStatus = false;
}

public boolean isEnabled ()
{
    return windowStatus;
}

```

```
////////////////////////////////////
// DESCENDANTS CAN OVERRIDE THESE METHODS
////////////////////////////////////
```

```
////////////////////////////////////
// myHandleEvent (evt)
//
// This will be the descendant's handleEvent() method, if it
// chooses to override it. If you do not handle an event,
// return false and this class will call super.handleEvent().
// If you do handle an event and do not want it to be passed
// up the component heirarchy then return true.
////////////////////////////////////
public boolean myHandleEvent (AWTEvent evt) throws Exception
{
    return false;
}
```

```
////////////////////////////////////
// eventNeedsAThread (evt)
//
// Override this method and return true if you want to spawn a
// thread to handle this event
////////////////////////////////////
public boolean eventNeedsAThread (AWTEvent evt) throws Exception
{
    return false;
}
```

```
////////////////////////////////////
// threadedHandleEvent (evt)
//
// This method gets envoked in its own thread if
// eventNeedsAThread(evt) returns true.
////////////////////////////////////
public void threadedHandleEvent (AWTEvent evt) throws Exception
{
}
```

```
////////////////////////////////////
// myAction (evt)
```



```

//
// This will be the descendant's action() method, if it
// chooses to override it. If you do not handle an event,
// return false. If you do handle an event return true.
////////////////////////////////////
public boolean myAction (AWTEvent evt) throws Exception
{
    return false;
}

////////////////////////////////////
// actionNeedsAThread (evt)
//
// Override this method and return true if you want to spawn a
// thread to handle this action
////////////////////////////////////
public boolean actionNeedsAThread (AWTEvent evt) throws Exception
{
    return false;
}

////////////////////////////////////
// threadedAction (evt)
//
// This method gets called in its own thread if
// actionNeedsAThread(evt,arg) returns true
////////////////////////////////////
public void threadedAction (AWTEvent evt) throws Exception
{
}

////////////////////////////////////
// preThreadInit ()
//
// This method is called immediately before a thread is spawned
// to deal with an action or event. The default behaviour is to
// disable the frame. Override this method if you wish to
// change that behaviour.
////////////////////////////////////
void preThreadInit ()
{
    setEnabled(false);
}

```

```

////////////////////////////////////
// postThreadCleanUp ()
//
// This method is called each time the thread handling events or
// actions is about to exit. The default behaviour is to enable
// the frame. Override this method if you wish to change that.
////////////////////////////////////
void postThreadCleanUp ()
{
    setEnabled(true);
}

////////////////////////////////////
// handleException (e)
//
// This method is called in its own thread when an
// ErrorException is raised by any of the event-handling methods.
// The default behaviour is to throw up an error window.
////////////////////////////////////
void handleException (Exception e)
{
    ErrorException error;

    // Convert the exception to an ErrorException if necessary
    if (e instanceof ErrorException) error = (ErrorException)e;
    else error = new ErrorException(e);

    // Do any initialization required before throwing up the error window
    preErrorWindowInit();

    // Make a new error window with the detail message specified
    // by the ErrorException
    ErrorWindow errorWindow = new ErrorWindow(error);
    errorWindow.addWindowListener(this);

    // Show the error window - this call blocks until the user
    // acknowledges the error by clicking the OK button
    errorWindow.show();

    // Dispose of the error window before returning
    errorWindow.dispose();

    // Do any cleanup required after disposing of the error window

```

```

    postErrorWindowCleanUp();
}

////////////////////////////////////
// preErrorWindowInit ()
//
// This method is called immediately before throwing up an error
// window. The default implementation disabled the frame.
////////////////////////////////////
void preErrorWindowInit ()
{
    setEnabled(false);
}

////////////////////////////////////
// postErrorWindowCleanUp ()
//
// This method is called immediately before throwing up an error
// window. The default implementation disabled the frame.
////////////////////////////////////
void postErrorWindowCleanUp ()
{
    setEnabled(true);
}
}

```

```

//////////
// Translation.java
//////////

import java.awt.Graphics;

class Translation
{
    int start, end;
    String text;
    String compressedText;

    static StringBuffer buf = new StringBuffer();

    void read (ReadStream in, char dirn) throws Exception
    {
        int i;
        String buf;
        Boolean status;
        StringBuffer bbuf = new StringBuffer();

        buf = in.readPrefix("[Translation = ");
        status = Boolean.valueOf(buf.substring(0,buf.length()-1));

        // If the translation is present then read it in
        if (status.booleanValue()) {
            in.readMessage("[Start Translation]");
            while (true) {
                buf = in.readLine();
                if (buf.equals("[End Translation]")) break;
                else bbuf.append(buf);
            }
            compressedText = bbuf.toString();
            bbuf.setLength(0);

            // Create the display string (reversing it if appropriate)
            if (dirn == 'r') {
                for (i=compressedText.length()-1; i>=0; i--)
                    bbuf.append(" " + compressedText.charAt(i) + " ");
            }
            else {
                for (i=0; i<compressedText.length(); i++)
                    bbuf.append(" " + compressedText.charAt(i) + " ");
            }
        }
    }
}

```

```

        }
        text = bbuf.toString();
    }

    else {
        doTranslation(dirm);

        // Create the output (compressed) string, reversing if appropriate
        if (dirm == 'r') {
            for (i=text.length()-2; i>=0; i-=3)
                bbuf.append(text.charAt(i));
        }
        else {
            for (i=1; i<text.length(); i+=3)
                bbuf.append(text.charAt(i));
        }
        compressedText = bbuf.toString();
    }
}

void draw (int xoffset, int fy, int ry, char dirm,
           float basesPerPixel, Graphics g)
{
    int x,tx;

    if (dirm == 'r') {
        int s = Grail.invertBaseNumber(end);

        if (s > xoffset) {
            x = (int)((s-xoffset)/basesPerPixel);
            tx = 0;
        }
        else {
            x = 0;
            tx = xoffset - s;
        }
        if (tx < text.length())
            g.drawString(text.substring(tx),x,ry);
    }
    else {
        if (start > xoffset) {
            x = (int)((start-xoffset)/basesPerPixel);
            tx = 0;
        }
        else {
            x = 0;

```

```

        tx = xoffset - start;
    }
    if (tx < text.length())
        g.drawString(text.substring(tx),x,fy);
    }
}

void doTranslation (char dirn)
{
    String gseq;
    int i;

    buf.setLength(0);
    buf.ensureCapacity(1+end-start);

    gseq = (dirn == 'f') ? Grail.sequence : Grail.rsequence;

    for (i=start-1; (i<end-2) && (i<Grail.sequenceLength-2); i+=3)
    {
        if (gseq.regionMatches(i,"TTT",0,3)) buf.append(" F ");
        else if (gseq.regionMatches(i,"TTC",0,3)) buf.append(" F ");
        else if (gseq.regionMatches(i,"TTA",0,3)) buf.append(" L ");
        else if (gseq.regionMatches(i,"TTG",0,3)) buf.append(" L ");
        else if (gseq.regionMatches(i,"CTT",0,3)) buf.append(" L ");
        else if (gseq.regionMatches(i,"CTC",0,3)) buf.append(" L ");
        else if (gseq.regionMatches(i,"CTA",0,3)) buf.append(" L ");
        else if (gseq.regionMatches(i,"CTG",0,3)) buf.append(" L ");
        else if (gseq.regionMatches(i,"ATT",0,3)) buf.append(" I ");
        else if (gseq.regionMatches(i,"ATC",0,3)) buf.append(" I ");
        else if (gseq.regionMatches(i,"ATA",0,3)) buf.append(" I ");
        else if (gseq.regionMatches(i,"ATG",0,3)) buf.append(" M ");
        else if (gseq.regionMatches(i,"GTT",0,3)) buf.append(" V ");
        else if (gseq.regionMatches(i,"GTC",0,3)) buf.append(" V ");
        else if (gseq.regionMatches(i,"GTA",0,3)) buf.append(" V ");
        else if (gseq.regionMatches(i,"GTG",0,3)) buf.append(" V ");
        else if (gseq.regionMatches(i,"TCT",0,3)) buf.append(" S ");
        else if (gseq.regionMatches(i,"TCC",0,3)) buf.append(" S ");
        else if (gseq.regionMatches(i,"TCA",0,3)) buf.append(" S ");
        else if (gseq.regionMatches(i,"TCG",0,3)) buf.append(" S ");
        else if (gseq.regionMatches(i,"CCT",0,3)) buf.append(" P ");
        else if (gseq.regionMatches(i,"CCC",0,3)) buf.append(" P ");
        else if (gseq.regionMatches(i,"CCA",0,3)) buf.append(" P ");
        else if (gseq.regionMatches(i,"CCG",0,3)) buf.append(" P ");
        else if (gseq.regionMatches(i,"ACT",0,3)) buf.append(" T ");
        else if (gseq.regionMatches(i,"ACC",0,3)) buf.append(" T ");
        else if (gseq.regionMatches(i,"ACA",0,3)) buf.append(" T ");
    }
}

```

```

else if (gseq.regionMatches(i,"ACG",0,3)) buf.append(" T ");
else if (gseq.regionMatches(i,"GCT",0,3)) buf.append(" A ");
else if (gseq.regionMatches(i,"GCC",0,3)) buf.append(" A ");
else if (gseq.regionMatches(i,"GCA",0,3)) buf.append(" A ");
else if (gseq.regionMatches(i,"GCG",0,3)) buf.append(" A ");
else if (gseq.regionMatches(i,"TAT",0,3)) buf.append(" Y ");
else if (gseq.regionMatches(i,"TAC",0,3)) buf.append(" Y ");
else if (gseq.regionMatches(i,"TAA",0,3)) buf.append(" * ");
else if (gseq.regionMatches(i,"TAG",0,3)) buf.append(" * ");
else if (gseq.regionMatches(i,"CAT",0,3)) buf.append(" H ");
else if (gseq.regionMatches(i,"CAC",0,3)) buf.append(" H ");
else if (gseq.regionMatches(i,"CAA",0,3)) buf.append(" Q ");
else if (gseq.regionMatches(i,"CAG",0,3)) buf.append(" Q ");
else if (gseq.regionMatches(i,"AAT",0,3)) buf.append(" N ");
else if (gseq.regionMatches(i,"AAC",0,3)) buf.append(" N ");
else if (gseq.regionMatches(i,"AAA",0,3)) buf.append(" K ");
else if (gseq.regionMatches(i,"AAG",0,3)) buf.append(" K ");
else if (gseq.regionMatches(i,"GAT",0,3)) buf.append(" D ");
else if (gseq.regionMatches(i,"GAC",0,3)) buf.append(" D ");
else if (gseq.regionMatches(i,"GAA",0,3)) buf.append(" E ");
else if (gseq.regionMatches(i,"GAG",0,3)) buf.append(" E ");
else if (gseq.regionMatches(i,"TGT",0,3)) buf.append(" C ");
else if (gseq.regionMatches(i,"TGC",0,3)) buf.append(" C ");
else if (gseq.regionMatches(i,"TGA",0,3)) buf.append(" * ");
else if (gseq.regionMatches(i,"TGG",0,3)) buf.append(" W ");
else if (gseq.regionMatches(i,"CGT",0,3)) buf.append(" R ");
else if (gseq.regionMatches(i,"CGC",0,3)) buf.append(" R ");
else if (gseq.regionMatches(i,"CGA",0,3)) buf.append(" R ");
else if (gseq.regionMatches(i,"CGG",0,3)) buf.append(" R ");
else if (gseq.regionMatches(i,"AGT",0,3)) buf.append(" S ");
else if (gseq.regionMatches(i,"AGC",0,3)) buf.append(" S ");
else if (gseq.regionMatches(i,"AGA",0,3)) buf.append(" R ");
else if (gseq.regionMatches(i,"AGG",0,3)) buf.append(" R ");
else if (gseq.regionMatches(i,"GGT",0,3)) buf.append(" G ");
else if (gseq.regionMatches(i,"GGC",0,3)) buf.append(" G ");
else if (gseq.regionMatches(i,"GGA",0,3)) buf.append(" G ");
else if (gseq.regionMatches(i,"GGG",0,3)) buf.append(" G ");
else buf.append(" A ");
}

```

```

text = (dirn == 'f') ? buf.toString() :
StringOp.reverse(buf.toString());

```

```

}
}

```

```

////////////////////////////////////
// WriteStream.java
//
// Next available error code: 36-3
////////////////////////////////////

```

```
import java.io.*;
```

```

class WriteStream
{
    private DataOutputStream out;

    WriteStream (FileOutputStream s)
    {
        out = new DataOutputStream(s);
    }

    WriteStream (DataOutputStream s)
    {
        out = s;
    }

    DataOutputStream getOutputStream ()
    {
        return out;
    }

    void close () throws IOException
    {
        try {
            out.close();
        }
        catch (IOException e) {
            throw new IOException(e,"Error closing stream",36,1);
        }
    }

    void writeBytes (String buf) throws IOException
    {
        int i;

        try {

```



```
        out.writeBytes(buf);
    }

    catch (IOException e) {
        throw new ErrorException(e,"Error writing to stream",36,2);
    }
}
```

4.2 Objective 2 -Parallel Database Search Algorithms

Develop both shared- and distributed-memory parallel implementations of the Smith-Waterman-Gotoh database search and comparison algorithms which will improve response time for searches and increase user capacity per server (with a goal of less dependence on specific architectures).

As the complete genomic sequence becomes available for more organisms, the role of supercomputers in the area of biological research will become more pervasive. The Smith-Waterman-Gotoh algorithm for matching biological sequences has long been accepted as an effective means of computer-based sequence analysis. We now describe the implementation used by the ApoCom GRAILTM software to perform the similarity search for repetitives in a sequence against a known database of repetitives¹³.

The Smith-Waterman-Gotoh (SWG) algorithm is an extension of the original pioneering work of Needleman & Wunsch (NW) for matching two proteins by the similarity in the amino acid sequences. One of the most attractive features of the NW algorithm is its stringency. Given two protein sequences, every amino acid in one sequence is compared to every amino acid in the other, thereby guaranteeing some measure of similarity. But as databases grow in size, such approaches become extremely time-consuming on all but the fastest computer systems.

The SWG algorithm has been shown to be well-suited to parallel processing by supercomputers. Before describing efficient implementations of the SWG method for both shared- and distributed memory supercomputers the algorithm will first be described. Although software exists to compare all four possible combinations (protein sequence against a protein database, protein sequence against a DNA sequence database, DNA sequence compared to a DNA sequence database, and a DNA sequence against a protein database), the following discussion will be for a protein sequence compared to a protein database.

The "smallest match" is a single amino acid from each protein sequence being compared. The term "maximum match" is the largest string of amino acids from the query protein sequence that can be matched from a string from a given protein sequence within the database allowing for all possible deletions/insertions (gaps) of any length. A scoring matrix is constructed having the scores ...

Let the two sequences under being compared be denoted by $X=x_1x_2x_3...x_N$ and $Y=y_1y_2y_3...y_M$. The basic problem of the SWG algorithm¹⁴ is to find the alignment of X and Y with the highest similarity. This alignment is accomplished in essentially MN steps. We define an alignment A of X and Y to be

¹³ Jurka J, Walichiewicz J, Milosavljevic A. J Mol. Evol. 1992, 35: 286-291.

¹⁴ Smith TF, Waterman MS. *Comparison of Biosequences*. Advances in Applied Mathematics 1981, 2:482-489.

Gotoh O, *An Improved Algorithm for Matching Biological Sequences*, J. Mol. Biol. 1982, 162:705-708.

$$A(X,Y) = \{(x_{i1},y_{j1}), (x_{i2},y_{j2}), \dots, 1 \leq i_1 \leq i_2 \leq \dots \leq N, 1 \leq j_1 \leq j_2 \leq \dots \leq M\}$$

Each x_i or y_j not appearing in the alignment is considered to be an insertion or deletion and will be denoted in the alignment by the symbol $\diamond$.

EXAMPLE: For the sequences $X=x_1x_2x_3x_4x_5x_6x_7$ and $Y=y_1y_2y_3y_4y_5y_6y_7y_8y_9$ one possible alignment is

$$A(X,Y) = \{(x_1,y_3), (x_3,y_4), (x_4,y_6), (x_6,y_7), (x_7,y_9)\}$$

This is depicted as

	$\diamond$		x_1	x_2	x_3	$\diamond$	x_4	x_5	x_6	$\diamond$	x_7
y_1	y_2	y_3	$\diamond$	y_4	y_5	y_6	$\diamond$	y_7	y_8	y_9	

Similarity measures are based on two weight functions. A distance weight $\alpha_k = d(x_m, y_n)$ function is assigned to an aligned pair x_i and y_j such that

$$d(x_m, y_n) \leq 0 \text{ if } x_m = y_n \quad \text{and} \quad d(x_m, y_n) > 0 \text{ if } x_i \neq y_j$$

This function measures the degree of similarity between two elements x_m and y_n , assigning the weight α_k to a match of type k . This algorithm allows multiple-sized gaps and assigns a gap weight $w_k \geq 0$ as a linear function of k to be the weight assigned to an insertion/deletion of length k :

$$w_k = u k + v \quad \text{where } u, v \geq 0$$

Then the similarity measure $S(X,Y)$ between X and Y is defined to be

$$S(X,Y) = \max_A (\sum \alpha_i \lambda_i - \sum w_k \Delta_k),$$

where λ_i is the number of matches of type i and Δ_k is the number of insertions/deletions of length k .

The SWG algorithm generates a distance matrix by induction in MN steps as follows:

$$D_{m,0} = P_{m,0} = 0 \text{ and } D_{0,n} = Q_{0,n} = 0$$

$$D_{m,n} = \min_{m,n} [D_{m-1,n-1} + d(x_m, y_n), P_{m,n}, Q_{m,n}]$$

where

$$\begin{aligned} P_{m,n} &= \min_{1 \leq k \leq m} [D_{m-k,n} + w_k] \\ &= \min [D_{m-1,n} + w_1, P_{m-1,n} + u] \end{aligned}$$

by recursion¹⁵

and

$$\begin{aligned} Q_{m,n} &= \min_{1 \leq k \leq n} [D_{m,n-k} + w_k] \\ &= \min [D_{m,n-1} + w_1, Q_{m,n-1} + u] \end{aligned}$$

by recursion.

A shared-memory parallel implementation of the SWG algorithm follows.

```
/*    GLOBAL DECLARATIONS    */

#include <stdio.h>
#include <string.h>
#include <math.h>

#define MAX_SEQ_LEN          6000
#define SEARCH_LEN           1000

#define MAX_SEQ_LEN_KEPT     1000
#define MAX_NUM_TOP_SEQ      100

#define MAX_NUM_TASKS        36
#define NPROC                 4

#define PATH_LEN              300
#define FILENAME_LEN          50
#define LINE_LEN              512
#define HEADER_LEN            85

#define MATRIX_DIM            24
```

¹⁵ Gotoh O, *ibid.*

```

#define      TMP_SPACE      50000

/* Modify for SWISS-PROT*/

#define VERY_BIG      44000
/* #define VERY_BIG      87000 */

#define ALPHA      -1
#define WEIGHT_1      (ALPHA + GAP)

#define MAX(a, b)      ((a) >= (b)) ? (a):(b))

      int      NUM_TOP_SEQ = 10;
      int      GAP = -10;

      int      chunksize, chunksize_last;

/* Modify with GRAILHOME */
      char      matrixbase[PATH_LEN] = "";
      char      matrixname[FILENAME_LEN] = "BLOSUM62";

/* Place database to be searched in shared memory for access by all
tasks */

struct database {
      char      access_num[HEADER_LEN];
      char      datab[MAX_SEQ_LEN];
      } dbs[VERY_BIG];

typedef struct {

      char      access[HEADER_LEN];
      int      score;
      char      *seq;

      } individual;

      individual seqstruct[MAX_NUM_TOP_SEQ*MAX_NUM_TASKS], tmpseq;

/* ***** Begin MAIN PROGRAM
***** */

main()
{

      FILE      *fileptr, *fptr;

      int      i, j;
      int      irec;

      int      accind[MAX_NUM_TOP_SEQ*MAX_NUM_TASKS];

```

```

    int    tmp_score[MAX_NUM_TOP_SEQ*MAX_NUM_TASKS];

    int    tem1, tem2;
    int    seqlen;

    char    path[PATH_LEN], line[LINE_LEN];

    char    *sq;

/* Modify with GRAILHOME */
    /*
    strcpy(path, "/home/grailgen/pro_db/sprot_flat");
    */
    strcpy(path, "sprot_flat");

    fileptr = fopen(path, "r");

    if (fileptr == NULL) {
        printf("ERROR EXIT: cannot open database file %s.\n", path);
        exit(1);
    }

/* Read the entire database into shared memory.
Each task will only search a portion of the database */

    irec = 0;

    while (    fgets(dbs[irec].access_num, HEADER_LEN, fileptr) != NULL) {
        fgets(dbs[irec].datab, MAX_SEQ_LEN, fileptr);
        irec = irec + 1;
    }

    fclose(fileptr);

/* DEBUG */
/*    printf(" irec = %d\n", irec); */
/*    fflush(stdout); */

/* Calculate size of database chunks for each task to process.
This calculation is based on the size of the database and the number
of tasks. */

    chunksize = irec / NPROC;

/* Calculate size of the last chunk in case it doesn't divide up
equally. */

    chunksize_last = chunksize + (irec % NPROC);

/* DEBUG
printf(" chunksize = %d, chunksize_last =
%d\n", chunksize, chunksize_last);
fflush(stdout);

```

```

*/

/* Begin the parallel region.
Each task compares only its portion of the database
which is stored in the structure dbs. */

/* Within the parallel region, the variables are distinguished
as follows:

#pragma parallel shared(chunksize,chunksize_last)

#pragma parallel byvalue(m,n) shared(query, subject, PAM)
*/
/* #pragma local(P,D,Q,D_prev,D_curr,temp,tem1,tem2,localmax)
*/

#pragma parallel

    {
        do_parallel();
    }

/* end of parallel region */

printf(" returned from parallel stuff \n");

/* sort top NUM_TOP_SEQ results from each of NPROC processors */

    for (i = 0; i < NUM_TOP_SEQ*NPROC; i++) {
        accind[i] = i;
        tmp_score[i] = seqstruct[i].score;

/*      DEBUG      */

        printf("accind    %d  tmp_score = %d\n", accind[i],tmp_score[i] );

    }

    for (i = NUM_TOP_SEQ*NPROC - 1; i > -1 ; i--) {
        for (j = 0; j < i; j++) {
            if (tmp_score[j] < tmp_score[j+1]) {
                tem1 = tmp_score[j];
                tmp_score[j] = tmp_score[j+1];
                tmp_score[j+1] = tem1;
                tem2 = accind[j];
                accind[j] = accind[j+1];
                accind[j+1] = tem2;
            }
        }
    }

    fptr = fopen("outfile", "w");
    for (i = 0; i < NUM_TOP_SEQ; i++) {
        fprintf(fptr, "score %5d %.40s\n", tmp_score[i],
seqstruct[accind[i]].access);

```

```

        /*printf("score %5d %.40s\n", tmp_score[i],
seqstruct[accind[i]].access);*/
        sq = seqstruct[accind[i]].seq;
        seqlen = strlen(sq);
        for (j = 0; j < seqlen; j += 60) {
            fprintf(fp, "%.60s\n", sq + j);
            /*printf( "%.60s\n", sq + j);*/
        }
    }
    fclose(fp);
}

/* *****End of MAIN PROGRAM *****
*/

/* ***** BEGIN the WORK Done in PARALLEL *****
*/

/* new routine:  do_parallel

each task is executing the search against a specific portion of the
database.  each task also writes out the score information which is
sorted by a post process

*/

    int      query[SEARCH_LEN];
    int      subject[MAX_SEQ_LEN];
    int      PAM[MATRIX_DIM][MATRIX_DIM];
    int      m,n;

do_parallel()
{
    FILE      *fileptr, *fp;

    char      accessnum[HEADER_LEN];

    char      tmp_work[TMP_SPACE], *q;

    char      msg_buf[SEARCH_LEN];

    char      work[MAX_SEQ_LEN], *p;

    char      path[PATH_LEN], line[LINE_LEN];

    int      i, j;
    int      minscore;

    int      minind, num_seq;
    int      tmp_score[MAX_NUM_TOP_SEQ];
    int      accind[MAX_NUM_TOP_SEQ];

```



```

int    my_id;

int    b_addr;          /* beg. address */
int    e_addr;          /* end address */

int    seq_com();

/*    int    query[SEARCH_LEN];    */

fileptr = fopen("query", "r");
if (fileptr == NULL) {
    printf("cannot open input file, quit.\n");
    exit(1);
}
fscanf(fileptr, "%s %d %d", matrixname, &GAP, &NUM_TOP_SEQ);
msg_buf[0] = '\0';
while (fscanf(fileptr, "%s", line) != EOF)
    strcat(msg_buf, line);
fclose(fileptr);
p = msg_buf + strlen(msg_buf) + 1;
sprintf(p, "%s", matrixname);
p += strlen(matrixname) + 1;
sprintf(p, "%d", GAP);
p += sizeof(int) + 1;
sprintf(p, "%d", NUM_TOP_SEQ);

printf("I rec %s %d %d seq_len = %d\n", matrixname, GAP,
NUM_TOP_SEQ, strlen(msg_buf));

/*    strcat(matrixbase, matrixname);
    fp = fopen(matrixbase, "r");    */
/* Modify with GRAILHOME */
fp = fopen("BLOSUM62", "r");
if (fp == NULL) {
    printf("cannot open MATRIX file %s, quit.\n", matrixbase);
    exit(1);
}

do {
    fgets(line, LINE_LEN, fp);
} while (line[0] == '#');

for (i = 0; i < MATRIX_DIM; i++)
    for (j = 0; j < MATRIX_DIM; j++)
        fscanf(fp, "%d", &PAM[i][j]);

fclose(fp);

transform(msg_buf, query);

m = strlen(msg_buf);

```

```

/* get this task's process id (my_id) */

my_id = mpc_my_threadnum();

printf(" my_id = %d\n",my_id);
fflush(stdout);

/* calculate beginning address in the database */

b_addr = 0 + (my_id * chunksize);
printf(" b_addr = %d\n",b_addr);
fflush(stdout);

/* calculate ending address */

if (my_id < (NPROC - 1))

    e_addr = b_addr + chunksize;
else
    e_addr = b_addr + chunksize_last;

printf(" e_addr = %d\n",e_addr);
fflush(stdout);

/* initialize some variables */

    minind = -1;
    num_seq = 0;
    q = tmp_work;
    p = work;

    for (i = b_addr; i < e_addr; i++) {

        strcpy(accessnum,dbs[i].access_num);
        /*printf(" accessnum = %s\n",accessnum);*/
        strcpy(p,dbs[i].datab);

        transform ( p, subject );

        n = strlen(p);

        tmpseq.score = seq_com();

        if (minind == -1 && num_seq == NUM_TOP_SEQ){
            minscore = 100000;
            for (j = 0 + my_id*NUM_TOP_SEQ; j < (my_id +
1)*NUM_TOP_SEQ; j++) {
                if (seqstruct[j].score < minscore) {
                    minscore = seqstruct[j].score;
                    minind = j;
                }
            }
        }
        if (num_seq < NUM_TOP_SEQ) {

```

```

        strcpy(tmpseq.access, accessnum);
        seqstruct[num_seq] = tmpseq;
        if (strlen(p) > MAX_SEQ_LEN_KEPT) {
            p[MAX_SEQ_LEN_KEPT - 1] = '\0';
        }
        strcpy(q, p);
        seqstruct[num_seq].seq = q;
        q += strlen(q) + 1;
        num_seq++;
    }
    else
        if (tmpseq.score > minscore) {
            strcpy(tmpseq.access, accessnum);
            seqstruct[minind] = tmpseq;
            if (strlen(p) > MAX_SEQ_LEN_KEPT) {
                p[MAX_SEQ_LEN_KEPT - 1] = '\0';
            }
            strcpy(q, p);
            seqstruct[minind].seq = q;
            q += strlen(q) + 1;

            minscore = 100000;
            for (j = 0 + my_id*NUM_TOP_SEQ; j < (my_id +
1)*NUM_TOP_SEQ; j++) {
                if (seqstruct[j].score < minscore) {
                    minscore = seqstruct[j].score;
                    minind = j;
                }
            }
        }
    }

}

/* ***** END the WORK Done in PARALLEL
***** */

/* ***** BEGIN seq_com routine ***** */

int seq_com()
{
    register int i, j;

    int Q, D_prev, D_curr, temp;
    int localmax = 0;

    int tem1, tem2;

    int D[MAX_SEQ_LEN];

```

```

    int      P[MAX_SEQ_LEN];

    extern max;

/* #pragma parallel if (n>111) byvalue(n) shared(P,D) local(i)
*/
/* #pragma pfor iterate(i=0;(n+1)/4;4) */

    for (i = 0; i <= n; ++i) {

        P[i] = 0;
        D[i] = 0;

    }

    for (i = 1; i <= m; ++i) {

        Q = 0;
        D_prev = D[0];
        D_curr = D[1];

        tem1 = query[i-1];

        for (j = 1; j <= n; ++j) {

            tem2 = subject[j-1];

/*            temp = D_prev + PAM[query[i-1]][subject[j-1]]; */
            temp = D_prev + PAM[tem1][tem2];

            P[j] = MAX (D_curr + WEIGHT_1, P[j] + ALPHA);

            Q = MAX ( D[j-1] + WEIGHT_1 , Q + ALPHA );

            D_prev = D[j];

/*            D_curr = MAX( P[j], temp );

            D_curr = MAX( Q, D_curr);

*/

            if ( temp < P[j])
                D_curr = P[j];
            else
                D_curr = temp;

/*            D_curr = MAX( D_curr, Q); */

            if ( D_curr < Q)
                D_curr = Q;

            D[j] = D_curr;

            localmax = MAX ( localmax, D_curr);

```

```

    }
}

return(localmax);
}

/* ***** END of seq_com ***** */

/* ***** BEGIN transform routine ***** */

int transformArray [] =
{
    0,      /* A */
    20,     /* B */
    4,      /* C */
    3,      /* D */
    6,      /* E */
    13,     /* F */
    7,      /* G */
    8,      /* H */
    9,      /* I */
    23,     /* J */
    11,     /* K */
    10,     /* L */
    12,     /* M */
    2,      /* N */
    23,     /* O */
    14,     /* P */
    5,      /* Q */
    1,      /* R */
    15,     /* S */
    16,     /* T */
    23,     /* U */
    19,     /* V */
    17,     /* W */
    22,     /* X */
    18,     /* Y */
    21,     /* Z */
    23      /* Outside range */
};

transform (const char *str, int arr[])

{
    register int  i, j, len;

    len = strlen (str);

    for (i = 0 ; i < len ; ++i )
    {
        j = str[i] - 'A';

```

```

        if ( j<0 || j>26)
            j = 26;

        arr[i] = transformArray[j];

    }
    /*    printf("gets here"); */
}
/*
transform(str, arr)
char *str;
int arr[];
{
    register int  i,  len ;

    len = strlen(str);

    for (i = 0; i < len; i++){
        switch (str[i]) {
            case 'A': arr[i] = 0;    break;
            case 'R': arr[i] = 1;    break;
            case 'N': arr[i] = 2;    break;
            case 'D': arr[i] = 3;    break;
            case 'C': arr[i] = 4;    break;
            case 'Q': arr[i] = 5;    break;
            case 'E': arr[i] = 6;    break;
            case 'G': arr[i] = 7;    break;
            case 'H': arr[i] = 8;    break;
            case 'I': arr[i] = 9;    break;
            case 'L': arr[i] = 10;   break;
            case 'K': arr[i] = 11;   break;
            case 'M': arr[i] = 12;   break;
            case 'F': arr[i] = 13;   break;
            case 'P': arr[i] = 14;   break;
            case 'S': arr[i] = 15;   break;
            case 'T': arr[i] = 16;   break;
            case 'W': arr[i] = 17;   break;
            case 'Y': arr[i] = 18;   break;
            case 'V': arr[i] = 19;   break;
            case 'B': arr[i] = 20;   break;
            case 'Z': arr[i] = 21;   break;
            case 'X': arr[i] = 22;   break;
            default: arr[i] = 23;
        }
    }
}
*/

```

4.3 Objective 3 - Frameshift Detection Algorithms

Develop and parallelize new database search algorithms for DNA-protein sequence comparison which detect multiple frameshifts in the DNA that effect translation. These algorithms can recognize very weak homologies between a translated DNA with multiple frameshifts and the SwissProt protein database with an accuracy and sensitivity of Smith-Waterman (at several times the computational cost). These algorithms will be especially valuable to customers who are doing single pass sequencing.

All experimentally-derived data is prone to error. A *frameshift mutation* results from addition or deletion of a base in the mRNA sequence. This can be illustrated using the following English sentence:

THE FAT CAT ATE THE BIG RAT

addition

THE FAT TCA TAT ETH EBI GRA T...

deletion

THE FAC ATA TET HEB IGR AT...

a second mutation in the reverse direction can cause reversion to the wild type (either at the exact position of the original mutation or a close position).

THE FAC ATA ATE THE BIG RAT

A sample output from a BLASTP run follows:

```
g8261.cand.3      1
MATFGAPATANSNPNSYEVTPSPADSISSLSFSRADILVATSWDNQVRCWEISRSGAS
sp|P40066|YET7_YEAST      ___DIA___QQ_FMFSA___GK_I_DV
sp|P41838|RAE1_SCHPO
___TSTVSN_TGDLK_DV___AQP_E___D_A___Q_EY_A_S___SK_IY_VQAT_Q_
sp|P20053|PR04_YEAST      ___DVKYH_SGKFIGSA_H_MTW_L_DA_
gi|886024          ___LG_TTS___P_LAIE_P_T_WVLDIA___DGSMVASG_R_GTA_L_NVA
gi|607003          ___VL_VA___ADGQRVASG_D_KTIKI_D

g8261.cand.3      61
LASAPKASISHDQPVLCSAWKDDGTTVFSGGCDKQAKMWPLLSSGGQPVTVMHEGPAAAM
sp|P40066|YET7_YEAST
R_QHESSS___TR_SN___KIVT_SW___TI_Y_DMRQPQPVS___M_P_A_KVL
sp|P41838|RAE1_SCHPO
IG_IG___LYE_QG___SVN_SR___K_AT_SW___TL_Y_D_RQSTPIA_SLP___VNSI
sp|P20053|PR04_YEAST
___KG_FLSFQC___SL_C___M_SLSML_DIR_SKVM_L_G_SK_YTV
gi|886024          LT_TE_HSV_FHPE___LA_ASE_GTIRI_IAT_KERDVLQAPAENVVSL
gi|607003          ___GGG_WSV_FSP___QR_A___SD_HTI_I_DAV___TCTQ_LEG_GDSVWSV
```

g8261.cand.3 121
 AWIPGMNLLATGSWDKTLKYWDTRQQNPVHTQQLPDKCYTLSVKHPLMVVGTAADRNLIVF
 sp|P40066|YET7_YEAST
 RFV_LYGTFFV_AGG_G_FNF_KN_RH_S_VMM_ERV_SMDN_QS_L_A_E_HIAH
 sp|P41838|RAE1_SCHPO
 _FH_QYGTFS_AGS_G_FSF_L_ST_IA_VS_ERV_AMDCV_LT_SV_KSAK_
 sp|P20053|PR04_YEAST _S_NGYQV_GG_GHNV_I_KRDEGQLN_I
 gi|886024 _FS_DGSM_VH_R_S_VHL_VASGEAL_
 gi|607003 _FS_DGQRV_S_I_G_I_I_ASGTCTQ_LNVGSTATC_FDYTNAYIN_

g8261.cand.3 181
 NLQNPQTEFKRIQSPLKYQTRCVTAFPDQQGFLACSLDAYILACGLRWALLSTLVHDI
 sp|P40066|YET7_YEAST _A_T_I_ATT_W_ACYNEAD_Y
 sp|P41838|RAE1_SCHPO DI_TG_NQ_LAM_F_SLAC_IKGD_Y

g8261.cand.3 241 LNIH

Protein match title lines

sp P40066 YET7_YEAST INTERGENIC	HYPOTHETICAL 40.5 KD PROTEIN IN SSA4-SWI4 score = 246 p value = 4.2e-55
sp P41838 RAE1_SCHPO polyA+ RNA	POLYA+ RNA EXPORT PROTEIN. gi 625094 (U14951) score = 244 p value = 5.8e-55
sp P20053 PR04_YEAST pir A32569	U4/U6 SMALL NUCLEAR RIBONUCLEOPROTEIN PRP4. score = 149 p value = 9.9e-13
gi 886024 curvata]	(U23820) protein kinase [Thermomonospora score = 114 p value = 6.2e-12
gi 607003 [Podospora anserina]	(I28125) beta transducin-like protein score = 134 p value = 5.7e-11

ApoCom worked on an algorithm which can detect and correct frameshift errors in DNA sequences during comparison of translated sequences with protein sequences in the databases. This algorithm is based on the SWG algorithm outlined in the last section. Timings for the SWG algorithm are given in the following figure. It is clear that the speedup is linear until you reach 8 processors, levelling off between 8 and 10 processors.

Figure 1. Wall Clock Time Versus Query Length

These timings were obtained from the Silicon Graphics Inc. Power Challenge redcurry in Boston, Massachusetts. A sample output for a query of length 320 for a 10-processor run is given below.

```
"big.8.320" [New file] 25 lines, 501 characters
```

```
redcurry 226% setenv MPC_NUM_THREADS 10
```

```
redcurry 227% timex ./ram10.exe
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
I rec BLOSUM62 -10 10 seq_len = 320
```

```
my_id = 9
```

```
my_id = 5
my_id = 6
my_id = 3
my_id = 4
my_id = 8
my_id = 7
my_id = 2
my_id = 1
my_id = 0
```

```
real 3:16.95
user 31:27.42
sys 22.81
```

The Job Control Language necessary to make these runs was as follows:

```
144 19:32 setenv
145 19:32 timex
146 19:35 timex ./ram10.exe
147 19:39 vi big.2.10
148 19:40 ls
149 19:41 cp query.160 query
150 19:41 setenv
151 19:41 dir big
152 19:41 ls
153 19:42 cat big.2.10
154 19:42 mv big.2.10 big.2.40
155 19:42 ls
156 19:43 rm core
157 19:43 vi big.10.40
158 19:43 ls
159 19:49 cat query
160 19:49 setenv
161 19:49 MPC_NUM_THREADS
162 19:49 setenv MPC_NUM_THREADS 8
163 19:50 timex ./ram8.exe
164 19:56 vi big.8.160
```

```

165 19:56 setenv
166 19:56 setenv MPC_NUM_THREADS 4
167 19:57 timex ./ram4.exe
168 20:01 vi big.4.160
169 20:02 setenv MPC_NUM_THREADS 4
170 20:02 redcurry 167% timex ./ram4.exe
171 20:02 I rec BLOSUM62 -10 10 seq_len = 160
172 20:02 I rec BLOSUM62 -10 10 seq_len = 160
173 20:02 I rec BLOSUM62 -10 10 seq_len = 160
174 20:02 I rec BLOSUM62 -10 10 seq_len = 160
175 20:02 my_id = 3
176 20:02 my_id = 0
177 20:02 my_id = 1
178 20:02 my_id = 2
179 20:02 real 3:39.31
180 20:02 user 14:11.79
181 20:02 sys 7.45
182 20:03 vi grabage
183 20:04 setenv
184 20:04 history
185 20:04 vi big.4.160
186 20:05 setenv MPC_NUM_THREADS 2
187 20:05 timex ./ram2.exe
188 20:14 170 20:02 redcurry 167% timex ./ram4.exe
189 20:14 171 20:02 I rec BLOSUM62 -10 10 seq_len = 160
190 20:14 172 20:02 I rec BLOSUM62 -10 10 seq_len = 160
191 20:14 173 20:02 I rec BLOSUM62 -10 10 seq_len = 160
192 20:14 174 20:02 I rec BLOSUM62 -10 10 seq_len = 160
193 20:14 175 20:02 my_id = 3
194 20:14 176 20:02 my_id = 0
195 20:14 177 20:02 my_id = 1
196 20:14 178 20:02 my_id = 2
197 20:14 vi big.2.160
198 20:15 ls -al big*
199 20:15 setenv
200 20:15 setenv MPC_NUM_THREADS 1
201 20:16 timex ./ram1.exe
202 20:49 8
203 20:49 vi big.1.160
204 20:49 ls -al spr*
205 20:50 ls -al que*
206 20:50 CP
207 20:50 cp query.320 query
208 20:50 setenv
209 20:50 setenv MPC_NUM_THREADS 2
210 20:51 timex ./ram2.exe

```

```

211 21:06 vi big.2.320
212 21:07 cp query.40 query
213 21:07 setenv
214 21:07 timex ./ram2.exe
215 21:10 rm big.2.40
216 21:10 vi big.2.40
217 21:11 setenv
218 21:11 cp query.320 query
219 21:12 setenv MPC_NUM_THREADS 4
220 21:12 timex ./ram4.exe
221 21:22 vi big.4.320
222 21:23 setenv
223 21:23 setenv MPC_NUM_THREADS 8
224 21:23 timex ./ram8.exe
225 21:27 vi big.8.320
226 21:28 setenv MPC_NUM_THREADS 10
227 21:28 timex ./ram10.exe
228 21:35 vi big.10.320
229 21:35 cp query.100 query
230 21:35 setenv
231 21:36 timex ./ram10.exe
232 21:39 vi big.110.10
233 21:39 vi query.100
234 21:40 cp query.110 query
235 21:40 cp query.110 query
236 21:40 cp query.100 query
237 21:40 timex ./ram10.exe
238 21:42 vi big.10.100
239 21:43 setenv
240 21:43 setenv MPC_NUM_THREADS 8
241 21:43 times ./ram8.exe
242 21:47 timex ./ram8.exe
243 21:49 history > big2
184 20:04 history
185 20:04 vi big.4.160
186 20:05 setenv MPC_NUM_THREADS 2
187 20:05 timex ./ram2.exe
188 20:14 170 20:02 redcurry 167% timex ./ram4.exe
189 20:14 171 20:02 I rec BLOSUM62 -10 10 seq_len = 160
190 20:14 172 20:02 I rec BLOSUM62 -10 10 seq_len = 160
191 20:14 173 20:02 I rec BLOSUM62 -10 10 seq_len = 160
192 20:14 174 20:02 I rec BLOSUM62 -10 10 seq_len = 160
193 20:14 175 20:02 my_id = 3
194 20:14 176 20:02 my_id = 0
195 20:14 177 20:02 my_id = 1
196 20:14 178 20:02 my_id = 2

```

197 20:14 vi big.2.160
198 20:15 ls -al big*
199 20:15 setenv
200 20:15 setenv MPC_NUM_THREADS 1
201 20:16 timex ./ram1.exe
202 20:49 8
203 20:49 vi big.1.160
204 20:49 ls -al spr*
205 20:50 ls -al que*
206 20:50 CP
207 20:50 cp query.320 query
208 20:50 setenv
209 20:50 setenv MPC_NUM_THREADS 2
210 20:51 timex ./ram2.exe
211 21:06 vi big.2.320
212 21:07 cp query.40 query
213 21:07 setenv
214 21:07 timex ./ram2.exe
215 21:10 rm big.2.40
216 21:10 vi big.2.40
217 21:11 setenv
218 21:11 cp query.320 query
219 21:12 setenv MPC_NUM_THREADS 4
220 21:12 timex ./ram4.exe
221 21:22 vi big.4.320
222 21:23 setenv
223 21:23 setenv MPC_NUM_THREADS 8
224 21:23 timex ./ram8.exe
225 21:27 vi big.8.320
226 21:28 setenv MPC_NUM_THREADS 10
227 21:28 timex ./ram10.exe
228 21:35 vi big.10.320
229 21:35 cp query.100 query
230 21:35 setenv
231 21:36 timex ./ram10.exe
232 21:39 vi big.110.10
233 21:39 vi query.100
234 21:40 cp query.110 query
235 21:40 cp query.110 query
236 21:40 cp query.100 query
237 21:40 timex ./ram10.exe
238 21:42 vi big.10.100
239 21:43 setenv
240 21:43 setenv MPC_NUM_THREADS 8
241 21:43 times ./ram8.exe
242 21:47 timex ./ram8.exe

243 21:49 history > big2
244 21:50 vi big.8.10
245 21:50 vi big.8.100
246 21:50 grep 320 big2
247 21:51 setenv
248 21:51 cp query.320 query
249 21:52 timex ./ram8.exe
250 21:56 vi big.8.320
251 21:57 cp query.100 query
252 21:57 setenv
253 21:57 cat big
254 21:57 s big*
255 21:57 ls big*
256 21:57 setenv
257 21:57 setenv MPC_NUM_THREADS 4
258 21:58 times ./ram4.exe
259 21:58 timex ./ram4.exe
260 22:01 ls
261 22:01 cp query.100 query
262 22:01 timex ./ram4.exetimex ./ram4.
263 22:02 timex ./ram4.exe
264 22:05 vi big.4.100
265 22:05 setenv
266 22:06 setenv MPC_NUM_THREADS 2
267 22:06 timex ./ram2.exe
268 22:11 vi big.2.100
269 22:12 setenv MPC_NUM_THREADS 1
270 22:12 timex ./ram1.exe
271 22:26 vi big.1.100
272 22:27 cp query.80 query
273 22:27 timex ./ram1.exe
274 22:39 grep > sprot_flat
275 22:39 grep ">" sprot_flat > aa
276 22:39 vi aa
277 22:39 ls
278 22:40 vi sprot_flat
279 22:40 ls -al
280 22:41 history
281 22:42 grep ">" sprot_flat_big > m
282 22:42 vi m
283 22:45 history >> big2

4.4 Objective 4 - User's Group and Technical Advisory Board

Form a GRAIL™-genQuest User's Group and a Technical Advisory Board. Conduct comprehensive surveys of commercial and Internet users to improve the visualization capabilities, functionality and GUI design for the display of sequence-level features, multiple organism comparison, and other identified needs which can be implemented during Phase II.

ApoCom formed a User's Group to beta test the software and to provide input for the development of the ApoCom GRAIL™ and the ApoCom Client Tool for Genomics (ACTG) software. Some of the Beta Test Sites included Abbott Laboratories, Darwin Molecular, Pangea Systems, the University of Texas, Genset SA, Sequana Therapeutics, SmithKline Beecham, Zymogenetics, Genetics Institute, and others. Most of these evaluators submitted written comments via Email which were implemented in the first release of the software products. The Beta Test Site Agreement used to protect the software follows.

BETA TEST SITE AGREEMENT

This BETA TEST SITE AGREEMENT (this "Agreement"), dated as of _____, 1996, is entered into by and between ApoCom Inc., (herein "Company"), a Corporation organized and existing under the laws of the State of Tennessee, and _____ ("Customer"), a _____ organized and existing under the laws of the State of _____.

Company and Customer, intending to be legally bound, hereby agree as follows:

Section 1

GENERAL

The "Beta Test," as further described herein, will last approximately three (3) months and will commence on or about _____, 1996. During the Beta Test, Customer shall be entitled to use the GRAIL Gene Characterization Server Software and the ApoCom Client Tool for GRAIL (the "Licensed Programs") for Customer's internal business needs and test and evaluation purposes only.

Section 2

RESPONSIBILITIES OF COMPANY

Company shall without charge for a period of ninety (90) days:

2.1 Furnish one (1) copy of the Licensed Programs in executable object code and associated documentation and one (1) copy of any modifications or additions produced in the Beta Test to Customer for trial use and testing in connection with the Beta Test.

2.2 Assist by telephone in installing the Licensed Programs at Customer's site in connection with the Beta Test.

2.3 Provide all corrections and/or enhancements to the Licensed Programs at no charge.

Section 3

RESPONSIBILITIES OF CUSTOMER

Customer shall:

3.1 Provide proper operating environment and suggested hardware per Company's specification in Exhibit A.

3.2 Install all fixes, releases, and updates to the Licensed Programs delivered to Customer.

3.3 Provide troubleshooting, debugging, and maintenance assistance during the Beta Test according to Company's request and detailed instructions.

3.4 Provide to Company such further documentation and assistance as may reasonably be required by Company to perform the Beta Test.

3.5 Use the Licensed Programs only as authorized herein, and instruct its employees having access to the Licensed Programs to act in conformity with Customer's obligations under this Agreement.

3.6 Refrain from making any modifications or additions to the Licensed Programs except with Company's written approval.

Section 4

OPERATING PROCEDURES - RESERVATIONS

4.1 Company will provide assistance to Customer for implementing and carrying out the Beta Test. However, Customer shall be responsible for the management, internal control, and implementation of the Beta Test, including acquiring adequate computer hardware, ensuring proper machine configuration and program installation, and assigning suitable personnel to the project.

4.2 Because the Licensed Programs may contain defects, Customer is responsible for establishing backup, log, batch, review, and other procedures and controls appropriate to maintain the integrity and continuity of Customer's operations.

4.3 Company is under no obligation to make any changes or modifications to the Licensed Programs suggested by Customer. Company reserves the right, with or without prior notice, to discontinue work on the Licensed Programs or to revise the Licensed Programs so it provides different features, features in different combinations, and/or different environment configurations.

Section 5

LIMITATIONS ON WARRANTY AND LIABILITY

5.1 THE LICENSED PROGRAMS PROVIDED IN THIS BETA TEST IS NOT A FULLY-TESTED PRODUCT OFFERING, AND HAS NOT BEEN COMPLETED ACCORDING TO COMPANY'S NORMAL DEVELOPMENT PROCEDURES. CUSTOMER'S USE OF THE LICENSED PROGRAMS IS AT CUSTOMER'S SOLE RISK; THE LICENSED PROGRAMS AND ASSOCIATED DOCUMENTATION MAY CONTAIN DEFECTS, FAIL TO COMPLY WITH APPLICABLE SPECIFICATIONS, AND PRODUCE UNINTENDED OR ERRONEOUS RESULTS WHEN OPERATED IN COMBINATION WITH OTHER COMPANY'S PRODUCTS. CUSTOMER ACCEPTS THE LICENSED PROGRAMS "AS IS." THE LICENSED PROGRAMS ARE PROVIDED AS IS, WITHOUT ANY WARRANTY WHATSOEVER.

5.2 IN NO EVENT SHALL COMPANY BE LIABLE FOR DAMAGES OF ANY KIND (DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL) RESULTING FROM ANY DEFECT IN THE LICENSED PROGRAMS OR ASSOCIATED DOCUMENTATION, EVEN IF IT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.3 CUSTOMER SHALL BE RESPONSIBLE FOR ANY USE OF THE LICENSED PROGRAMS IN CUSTOMER'S OPERATIONS. CUSTOMER SHALL BE RESPONSIBLE FOR VERIFYING ANY OUTPUT RESULTING FROM USE OF THE LICENSED PROGRAMS IF CUSTOMER INTENDS TO USE OR RELY ON SUCH OUTPUT FOR BUSINESS PURPOSES. CUSTOMER SHALL FOLLOW PROPER BACKUP PROCEDURES FOR ANY OTHER PROGRAMMING AND ALL DATA TO PROTECT AGAINST LOSS OR ERROR RESULTING FROM USE OF ANY OR ALL OF THE LICENSED PROGRAMS.

Section 6

OWNERSHIP AND CONFIDENTIALITY

6.1 Company shall have sole and exclusive ownership of all right, title, and interest in and to the Licensed Programs and any additions or modifications thereto arising during the Beta Test or as a result of any other collaboration between Company and Customer and all intellectual property rights associated therewith (including, without limitations, rights to copyrights, trade secrets, or know-how). Upon

Company's request, Customer agrees to execute such further instruments and take such further action as Company may reasonably request to effect ownership of any such modifications.

6.2 Customer agrees to maintain in confidence the Licensed Programs and all documentation that relate to the design, development, operation, testing, or use of the Licensed Programs or any additions or modifications thereto. Customer agrees to use the same security measures as it uses in its business practice to protect its own confidential technical information. Customer further agrees not to disclose such information to anyone other than those of its employees, under nondisclosure obligations, who have a need to know such information in connection with the Beta Test. Upon completion of the Beta Test, unless Customer and Company enter into a further signed, written license agreement for the extended use of the Licensed Programs, Customer agrees to return all the License Programs and associated documentation to Company and Customer further agrees to certify its compliance with such obligation if requested by Company. The foregoing confidentiality obligations shall not apply to any information generally available to the public, independently developed or obtained without reliance on Company's information, or approved for release by Company without restriction.

6.4 Customer acknowledges that the GRAIL Gene Characterization Software, one of the Licensed Programs furnished by Licensor in the performance of this Agreement, constitute Licensed Materials that are proprietary and confidential to the Martin Marietta Energy Systems, Inc. (now Lockheed Martin Energy Research Corporation) and that Company has such materials lawfully in its possession as a Licensee. Customer agrees that it will not remove any notices of copyright or governmental sponsorship placed upon such Licensed Materials. Further, Customer acknowledges that the ApoCom Client Tool for GRAIL is a copyrighted work of ApoCom and agrees that it will not remove any notice of copyright placed upon such Licensed Program.

Section 7

TERMINATION

7.1 Either party may terminate the Beta Test and this Agreement at any time.

7.2 Upon expiration or termination of the Beta Test for any reason, provided that Customer has not entered into a License Agreement with Company on mutually agreeable terms for the ongoing use of the Licensed Programs, Customer shall immediately return all copies of the Licensed Programs, and associated documentation that relate to the design, development, operation, testing, or use of the Licensed Programs or any additions or modifications thereto. Customer agrees to certify its compliance with the foregoing promptly upon Company's request.

Section 8

MISCELLANEOUS

8.1 The provisions of Sections 4, 5, and 6 shall survive termination of this Agreement.

8.2 This Agreement shall be governed by and construed in accordance with the laws of the State of Tennessee, without regard for principles of choice of law.

8.3 This Agreement together with the attached Exhibits A constitute the complete and exclusive statement of the agreement of the parties with respect to the subject matter hereof, and supersede all prior oral and written proposals, representations, or other communication related to the subject matter hereof. The Licensed Programs and the actions and commitments of Company with respect thereto shall not be construed as being directly or indirectly covered by or subject to any other agreement between Customer and Customer.

IN WITNESS WHEREOF, the undersigned parties have caused, under seal, their authorized representatives to execute and deliver this Agreement as effective the date and year first above written.

ApoCom Inc.

By: _____
Ruth Ann Manning, Ph.D.

Title: President

Date: _____

[Customer] _____

By: _____
(name)

Title: _____

Date: _____

EXHIBIT A

SOFTWARE: GRAIL Software System [including 1) the **GRAIL** command-line version; 2) **Batch GRAIL** for automated processing of small DNA fragments (e.g., cDNAs, ESTs, cosmid skimming); 3) the interactive client/server version with the Java™-based **ApoCom Client Tool for GRAIL**; and 4) complete documentation for installation and usage.]

Authorized Server Computer System: Architecture and Model _____
Serial Number _____, Host ID _____
(e.g., SUN, SGI, DEC, IBM)

Authorized Client Computer Systems: _____
(e.g., PC (Windows® NT, Windows® 95), UNIX, Macintosh)

Media: _____ (e.g., 4mm or 8mm tape)

Server Platform/Operating System: _____ (e.g., SolarisVersion 2.x)

Network Type: TCP/IP

License Fee: Free of charge for Term of this Agreement

Payment Terms: N/A

Notices:

	COMPANY	CUSTOMER
To:	Ruth Ann Manning	_____
Address:	ApoCom Inc.	_____
	1020 Commerce Park Drive	_____
	Suite F	_____
	Oak Ridge, TN 37830	_____
Telephone:	423-482-2500	_____
Fax:	423-220-2030	_____

Copy to:	Paulette F. Rice	_____
Telephone:	423-482-2500	_____
FAX:	423-220-2030	_____

ApoCom formed a Technical Advisory Board made up of industry leaders who met once in Oak Ridge, Tennessee, twice in Hilton Head, SC at the TIGR Genome Sequencing and Analysis Conferences, and consulted with ApoCom on an on-going telephone and Email basis throughout the life of this contract. Many of the Technical Advisory Board members were also Beta Testers of the software and provided much valuable inputs for its development. Ruth Ann Manning, Richard Gibbs, Dick McCombie and Bruce Roe were speakers at the special meeting on Gene Finding held at the Banbury Center at Cold Spring Harbor, NY in March, 1997.

ApoCom Technical Advisory Board

- **Lydie Bougueleret, Ph.D.** **GENSET SA, France**

GENSET is a public biotechnology company engaged in the systematic and comprehensive analysis of the human genome. Their mission is to identify and patent genes and regulatory regions related to selected common diseases (including prostate cancer, schizophrenia, and osteoporosis), to discover drugs for such diseases, and to enter into strategic partnerships with pharmaceutical companies which will develop and market these drugs. GENSET has developed the NetGene database and Biointelligence analysis software.

- **Richard Gibbs, Ph.D.** **Baylor College of Medicine**

Richard is the **Director** of the **Baylor Human Genome Sequencing Center**. The research from his laboratory is focused in three main areas: 1) polymerase chain reaction (PCR) methods for studying the molecular basis of human X-linked genetic diseases; 2) the molecular basis of somatic mutations in mammalian cells; and 3) direct sequencing of the human immunodeficiency virus (HIV).

- **Jim Golden, Ph.D.** **GeneSys Inc.**

Jim is recognized as a leading expert in the field of neural networks and DNA sequence analysis software development. He also consults for ApoCom on our SBIR Phase II Grant. GeneSys develops instruments for automated DNA sequencing.

- **Jim Lindelien** **Time Logic, Inc.**

Jim is the founder and President of Time Logic. Time Logic's **DeCypher II** is a scalable, high-performance array processor hardware technology which can be added to Unix, Mac, and PC workstations to accelerate Smith-Waterman and other popular genetic similarity algorithms to speeds well in excess of present supercomputers.

- **Dick McCombie, Ph.D.** **Cold Spring Harbor Laboratory**

Cold Spring Harbor Laboratory is an independent, nonprofit research and educational institution chartered by the University of the State of New York. When James Watson became the Director in 1969, he focused the Laboratory's research on the genetics of cancer, using viruses as a model system. Dick McCombie, Richard Gibbs, Ellson Chen (Perkin Elmer Corporation), and Richard Wilson (Washington University) team each year to teach a two-week course on recent advances in high speed automated DNA sequencing. The Cold Spring Harbor *Arabidopsis* Genome Sequencing Project uses genetic approaches coupled with the high throughput sequencing of genomic DNA and cDNA tags from selected libraries. Cold Spring Harbor is also committed to the characterization of the fission yeast *Schizosaccharomyces pombe* genome.

- **Yannick Pouliot, Ph.D.** **Molecular Simulations, Inc.**

Molecular Simulations provides computational chemistry software and services to organize scientific data, share information, and develop, analyze, and simulate novel compounds and processes. Specific tools are provided for analytical instrumentation, catalysis and sorption, chemical reactions, crystallization, drug design, molecular biology, NMR, polymer modeling, protein design, quantum mechanics, software development, and X-ray crystallography. Yannick is now with Progenitor.

- **Bruce Roe, Ph.D.** **University of Oklahoma**

Bruce is the **Director** of the **Advanced Center for Genome Technology**, located on the Norman Campus of the University of Oklahoma. This Center is a state-of-the-art DNA sequencing facility, having completed the sequence of greater than three million bases of human genomic DNA from cosmid, P1 and BAC clones from human chromosome 22.

- **Eric Snyder, Ph.D.** **AxyS Pharmaceuticals (formerly
Sequana Therapeutics Inc.)**

Eric and his major professor G.D. Stormo (University of Colorado) developed a well-known neural networks software package called **GeneParser** which is similar to **GRAIL™ GeneParser** identifies and determines the fine structure of protein genes in genomic DNA sequences. Eric is also a consultant to ApoCom for the SBIR Phase II Grant. Sequana Therapeutics is searching for disease genes, including type II diabetes, asthma, osteoporosis and schizophrenia. The identification of disease genes will provide insights into the causes of common diseases and will facilitate the development of highly specific diagnostic and therapeutic products, such as small molecule drugs, recombinant proteins, gene therapy and antisense therapy.

- **Clark Tibbetts, Ph.D.** **Vanderbilt University**

Clark is a Professor of Microbiology at Vanderbilt University. He is a recognized leader in the field of neural networks. Clark has developed a contextual basecalling system which operates as a pipeline of signal conditioning algorithms and neural network data

stream transformations. His research in the area of high speed DNA sequencers using arrays of microchannels etched in a glass substrate is leading the edge of high throughput DNA sequencing technologies.

4.5 Objective 5 - New Strategies

Develop new computer tools, information systems, and strategies for the annotation, management, analysis, interpretation, and dissemination of large amounts of genomic mapping and sequencing information..

ApoCom researched the possible use of fuzzy systems for gene prediction and gene structure investigation. Fuzzy systems were used to model functions that help us predict the location of significant features in nucleotide sequences.

It is possible for a fuzzy system (as defined in Section 3.3) to compute any function over the set of real numbers of dimension n into dimension m . However, in practice, fuzzy systems are usually used in a restricted form that facilitates a more intuitive rule-based representation. In the restricted form, fuzzy systems cannot capture all functions over the set of real numbers. However, a restricted fuzzy system with $n+1$ rules can be used to approximate an n th order polynomial.

Since fuzzy systems in the restricted form have a very intuitive rule-based representation, they have two main advantages over more opaque function-modeling techniques such as artificial neural networks and Markovian methods. Specifically, it is possible to obtain an "explanation" of predictions made by a fuzzy system by examining the rules it used in making these predictions. In addition, an examination of the entire set of rules may help us to identify the specific biochemical mechanisms that determine the location of significant genome features.

ApoCom's new software GeneHound™ can be trained for any organism, thereby removing the restriction to humans that the ApoCom GRAIL™ software requires. ApoCom can also offer a dynamic programming software that produces Smith-Waterman sequence homology results on low-end workstations at rates that exceed parallel supercomputer performance.

4.6 Objective 6 - GRAIL™ Improvements

Improve GRAIL™'s technology for the identification of all coding sequences, genes, and other functional elements in genomic DNA (from both non-human organisms and the human).

ApoCom made several improvements to the basic GRAIL™ software in response to the requests from its User's Group, Technical Advisory Board, and customers. The ApoCom Client Tool for Genomics (ACTG) provides a rich and intuitive graphical representation of coding regions, RNA Pol II promoters, gene models, coding strand determination, translation start sites, CpG islands, gene model translation, local GC content, repetitive elements and polyadenylation sites.

MAIN WINDOWS

Initially only the File Menu is enabled, since a sequence must be loaded before any other actions can be taken. After a sequence has been loaded and analyzed using the Grail 2 algorithm, ACTG displays four main windows.

Table 1. Main Windows for ACTG™

<i>Window</i>	<i>Function</i>
ACTG Window	Displays a graphical representation of the ApoCom GRAIL™ analysis of the input sequence.
DNA SEQUENCE Window	Shows a segment of the DNA sequence loaded from the sequence file.
ANALYSIS Window	Displays tabular information about exons and gene models found in the sequence.
FEATURES Window	Displays tabular information about the polyadenylation sites, Pol II promoters, CpG islands, and repetitive elements found in the sequence.

4.6.1 ACTG Window

This window contains graphical representations of the exons, gene models, polyadenylation sites, promoter regions, CpG islands, and repetitive DNA elements. When you want to select an individual feature for further analysis, you simply click on its graphical representation within this window. A gray-scale representation of the GC

content and graphical representations of Shadow Exons¹⁶ can be toggled ON or OFF in this window using the Features Menu.

4.6.1.1 Loading a Sequence or Pre-Analyzed Sequence File

The first step in using the **ACTG Interactive Tool for ApoCom GRAIL™** is to load a DNA sequence file or a sequence that has previously been analyzed. A sequence file is simply a text file in FastA format containing the DNA sequence under consideration.

- An optional sequence name can be placed in the file before the sequence and preceded by a "greater than" (>) symbol.
- The length of a sequence must be at least 100 bases (for **Grail 1**). There is no imposed maximum length.
- The sequence must consist of upper or lower case letters A, C, G, T or U (a, c, g, t, or u).
- U is converted to T.
- Any other character is converted to C except blank lines, spaces, and digits that are ignored.
- **ApoCom GRAIL™** can read sequence files spaced into human-readable formats with sequence base numbers included.

Selecting the menu option **Load** from the File Menu causes an Open dialog box to pop up for accessing all subdirectories and files.

- If a raw sequence file is selected, then the sequence is read from the file and a **Grail 2** analysis is performed. Depending on the size of the sequence, it may take a few minutes for the results to be returned.
- If a pre-analyzed file is opened, all of the results from the previous analysis are available from the four main windows.

¹⁶ If the same exon is recognized on both strands of the DNA, the exon candidate on the more likely strand is called "Standard." A rejected exon is labelled "Shadow."

Using the ACTG Window File Menu, **LOAD** the sample DNA sequence file (**SAMPLE1.seq**) included with your ApoCom **GRAIL** Personal Edition software. Part of this sequence is shown in **Figure 3.1**. This sequence represents 62,557 base pairs of a human DNA cosmid clone.

```
>SAMPLE1
tcgacgggatcacgaggcaggagatcgagaccatcctgactaccacgggtgaaccccgctctactaaaaaatacaaaaaactagccaggcatggg
gogggcgctgtagttccagctactcgggaggctgaggaggaggagaatggcgtgaacccgggaggcggagcctgagtgagccgagattgtccactgc
actccagcctgggcgacagagcgagactccgtctcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagaaaaggattgtaagagtactgtta
cattttctggcctactacccttaaaattcgtgtgcaattctttgtatttacaaggaaaagactgaactttttctcatcaaaactagctttttctcacagggtaaactgc
accaatgtctgctcttttttaattgttttgtaactctgggcagactcagtttttaaaaaataaagattctaagcagctatcttggcattcccttaatactgtct
taacctcctacttttatttccactccttccacacacatgcataaatcctttacottttaagaatcattaagactgtcacacattaggaaactcttgcctcactctc
tgtcatttgctgcaatattgaaattctattttgaccatcaatgccatttaattcttctaatacatgaagaaaatgattgagtagcagcagtactatagggtggaaata
cagtttaactgctgaattttatacctctctgatttatagcttgctaattaattgctattaatagtttggttgcttaattagacttaagaaaacaacagggttttttttg
agggttttttttttttgcatgaggagagaattgtatgaaccagtgatgattatcctgaatgtacagacagaagtaagcctggacattgttaatttaaaaact
ttagtccttgcttaagggaatatgataatgtatactatgacaaatgtactttattcttctaacacagtaagaattacttgaactttttctgaaactaagtgcagga
aagccctgtgtgtcttgggttagtgatggttcaattcta
```

Figure 2. Partial Sample Sequence File (SAMPLE1.seq)

4.6.1.2 A Closer Look

Once an analysis is available for the current sequence under consideration (whether read in from an existing analysis file or computed by **ApoCom GRAIL™**), all menus and controls are enabled in the ACTG Window (see Figure 3).

Each of the ACTG Window menus and their options will be discussed below. The central horizontal, gray-scale band in the center of the ACTG Window shows the GC content of the sequence. The lighter the band, the higher the concentration of G (guanine) and C (cytosine).

Other portions of the **ApoCom GRAIL™** analysis can be seen by using the Scrollbar (faster) or Arrows (slower) located at the bottom of the ACTG Window. The initial width of the ACTG Window depends upon your display (usually about 5kb wide). Longer sequences can be fit into the window by using the Zoom features.

ApoCom GRAIL™ performs its analysis of the sequence in the order it was presented in the sequence file (Forward) as well as on the complementary nucleotide sequence (Reverse). The DNA sequence coding recognition analyses identifying potential coding exons on the Forward and Reverse strands are shown above and below the central band, respectively, as a line graph representing the results of the neural network recognition.

Above certain line graph peaks (below on the reverse strand) are cyan bars identifying the probable exons. Shadow Exons are represented by gray bars and can be toggled ON and OFF using the Features Menu. Gene models will also be represented in the ACTG Window by a set of linked yellow bars positioned above (below) the cyan bars.

Figure 3. ApoCom GRAIL™ Personal Edition Main ACTG Window
(Reverse Video Selected, Print Frame Selected)

4.6.2 DNA SEQUENCE Window

This window displays a fixed-size base segment of the DNA sequence from the (Forward) as well as on the complementary (Reverse) nucleotide sequence strands (see Figure 4). The protein translation of exons can be displayed within the DNA SEQUENCE Window. The exon for translation can be selected by either clicking on the chosen line in the ANALYSIS Window's GRAIL™ EXON TABLE or clicking on the corresponding color bar in the ACTG Window or DNA SEQUENCE Window. In either case, the selected exon's color bar will be enclosed in a yellow box in both the ACTG and DNA SEQUENCE Windows. The protein translation for the selected exon will be displayed in the central portion of the DNA SEQUENCE Window. For a discussion of the **Print** and **Save Image** options in the DNA SEQUENCE Window File Menu, see the section describing the ACTG Window File Menu.

GCTTCACCACCACGGCCGAGCGGGAAATCGTGCGCGACATCAAGGAGAAGCTGTGCTA
S F T T T A E R E I V R D I K E K L C Y

CGAAGTGGGTGGTGCCGGCTCGCCCTTTAGCACGCGCTGTAGTTCTTCTCGACACGAT

**Figure 4. DNA SEQUENCE Window
(Reverse Video Selected, Print Frame Deselected)**

Move through the sequence by clicking on the Scrollbar (faster) or Arrows (slower) at the ends of the Scrollbar in the DNA SEQUENCE Window.

Find a candidate exon in the ACTG Window and select it by clicking on its color bar. The output shown in the DNA SEQUENCE Window is automatically positioned to display a portion of this exon's sequence. Notice that the DNA SEQUENCE Window highlights the potential exons with a color-coded horizontal bar corresponding to what is shown in the ACTG Window.

Choose an exon from the ACTG Window and select it by clicking on its color bar.

Find the chosen exon's entry in the GRAIL EXON TABLE in the ANALYSIS Window. This line will be highlighted.

The DNA SEQUENCE Window will automatically be positioned to display the sequence of your chosen exon. Notice the cyan letters (gray letters for Shadow Exons) in the central portion of this window corresponding to the exon's protein translation.

Click on the entry in the GRAIL EXON TABLE above or below your chosen exon. This new line and the corresponding color bar are highlighted in the ACTG Window. The new exon and the protein translation for the new exon are now displayed in the DNA SEQUENCE Window.

4.6.3 ANALYSIS Window

This window contains three tables that contain information about the GRAIL Exons, Model Exons, and Gene Models displayed graphically in the ACTG and DNA SEQUENCE Windows. Clicking on the appropriate tab (GRAIL Exons, Model Exons, or Gene Models) causes the corresponding table of information to be brought to the front of the display.

The Gene Model Translation will be displayed in the bottom portion of the GENE MODEL TABLE in the ANALYSIS Window. Stop codons will be denoted by an asterisk ("*"). This information and all other tabular information is accessible from a "Cut-and-Paste to Clipboard" feature for easy export of this data to your annotation and other reports. For a discussion of the File Menu **Print** and **Save Image** options and the Copy Menu **Entire List**, **List Entry**, and **Translation** options for the ANALYSIS Window, see below for the corresponding options in the FEATURES Window.

ANALYSIS: Sequence: SAMPLE1 Length: 62557				
GENE MODEL TABLE ☒			# of Models: 4	
Str	Seq	Range	Score	Exons
f	2	62557	2030	27
r	1	62556	482	7
f	19000	22750	-54	2
r	39808	43558	405	5
HAVEFDGGVWVGSDSRVSAGEAVVNRVFDKLSPLHERIYCALSGSAADAQAVADMAAYQLELHGIELEEFFLVLAANNWR				

Figure 5. GENE MODEL TABLE in ANALYSIS Window
(Print Frame Selected)

4.6.3.1 GRAIL EXON TABLE

This subwindow displays information for each of the exons found by the **ApoCom GRAIL** coding recognition facility. The columns contain the **Type** of exon (Standard or Shadow), strand (Forward or Reverse Strand) from which the exon is taken (**Str**), reading frame (**Fr**), position of the exon in the sequence (**Position**), limits between which the preferred reading frame is open (**Op Rd Frm**), and quality score (**qual**).

The example SAMPLE1 has 41 Standard Exons identified (59 including Shadow Exons). You can scan through them using the Scrollbar on the right-hand side of the table.

Notice that the first 33 identified exons are from the Forward strand. Six of these (exons 2, 11, 18, 20, 28, and 33) are rated as "good," four (exons 1, 3, 12, and 19) are "marginal," while the remaining twenty-three are "excellent." The other eight Standard Exons within this table are from the Reverse strand.

The Grail 2 algorithm identifies 18 Shadow Exons (3 on the Forward Strand). These are shown in gray in the ACTG and DNA SEQUENCE Windows.

4.6.3.2 MODEL EXON TABLE

After Gene Models have been assembled, this subwindow displays information for each of the Model Exons that make up the currently-selected Gene Model. The columns contain the **Type** of exon (Standard or Shadow), strand (Forward or Reverse Strand) from which the exon is taken (**Str**), reading frame (**Fr**), position of the exon in the sequence (**Position**), and a quality score (**Score**) of the relative measure of the confidence of the neural net recognition system (with an upper limit of 100 being the best).

4.6.3.3 GENE MODEL TABLE

This subwindow displays information for each of the Gene Models assembled by the **ApoCom GRAIL Personal Edition**. The columns contain Forward or Reverse strand (**Str**), region of the sequence considered in assembling the model (**Seq Range**), score (**Score**), and the number of exons in the model (**Exons**). The **Score** is a relative measure of the GRAIL dynamic programming model confidence. The higher the score between similar models, the better the model is deemed to be by GRAIL.

4.6.4 FEATURES Window

The FEATURES Window displays tabular information about the polyadenylation sites, Pol II promoters, CpG islands, and repetitive elements.

FEATURES: Sequence: SAMPLE1 Length: 62557				
☐ PolyA Site	Start	End	Score	%GC
☐ Promoters	2543	2768	0.81	51.63
☐ Promoters	24060	25188	0.74	65.70
☒ CpG Island	25395	25597	0.61	66.15
☐ CpG Island	34585	34945	0.75	65.68
☐ Reptiv DNA	39745	40051	0.64	55.27

Figure 6. CpG Island Table in the FEATURES Window
(Print Frame Selected)

4.6.4.1 PolyA Sites

This subwindow displays information for each of the polyadenylation sites found by ApoCom GRAIL. Each site is reported in the following format:

Str Start End Score

The columns contain the strand (Forward or Reverse Strand) on which the PolyA Site is located (**Str**), its position within the sequence (**Start** and **End**), and a relative measure of the GRAIL neural network recognition confidence (**Score**).

4.6.4.2 Promoters

This subwindow shows a table of Pol II promoter regions that were located. Each region is reported in the format:

Str Score CAAT GCbox TATA Cap ATG

where **Str** is the strand ("f" for forward or "r" for reverse), **Score** is a relative measure of the GRAIL neural network recognition confidence, **CAAT** gives the location of the CAAT box, **GCbox** gives the location of the GC box, **TATA** gives the location of the TATA box, **Cap** gives the location of the cap site, and **ATG** gives the location of the translation start site. If any of these fields is reported as "-1," it means that no such site was found.

4.6.4.3 CpG Islands

This subwindow reports the CpG islands that were located by **ApoCom GRAIL Personal Edition**. Each entry is reported in the following format:

Start End Score %GC

where **Start** and **End** give the location of the site, **Score** is a measure of the site's quality, and **%GC** is the percentage of GC content.

4.6.4.4 Repttv DNA

This subwindow reports repetitive DNA items. Each entry has the following format:

Type Str Start End Score

where **Type** gives the type of the repetitive element, **Str** is the strand ("f" for forward, or "r" for reverse), **Start** and **End** give the location of the element, and **Score** is the score computed by the Smith-Waterman algorithm to measure the alignment of repetitive DNA elements to the input sequence.

4.6.5 The ACTG Window Menus

4.6.5.1 File Menu

Any time during an ACTG session, you have the option to save the current state of analysis to an analysis file by selecting the **Save** option from the File Menu. A dialog box is displayed allowing you to specify the output file name. Selecting "Complementary Mode" from this dialog box allows you to save regions on the reverse strand as the complement of regions on the forward strand.

NOTE: If you try to **Load** another sequence during an analysis session, you are first prompted with the **Save** option dialog box.

ACTG also provides you with the ability to print the information you see on your screen or to save this information as Graphics Interchange Format (GIF) images. ApoCom has entered into a GIF Software Patent License Agreement with UNISYS CORPORATION¹⁷ to use their patented LZW compression technology for the GIF file format.

¹⁷ For information concerning licensing the proprietary LZW technology, please contact UNISYS CORPORATION, Welch Licensing Department, M/S C1SW19, Township Line & Union Meeting Roads, P. O. Box 500, Blue Bell, Pennsylvania 19424.

NOTE: This Agreement with UNISYS CORPORATION allows you the GIF capability for use **ONLY** on a single personal computer that is not used as a server.

Selecting the **Save Image** option allows you to save the image in the ACTG Window in four distinct forms:

- **Default:** As you see the image on the screen, but without the ACTG Window's frame.
- **Reverse Video:** The black background of the image is changed to white.
- **Print Frame:** The ACTG Window's frame is saved.
- **Reverse Video and Print Frame:** White background and ACTG Window's frame saved.

Selecting the **Print** option in the File Menu pops up the usual Print dialog box and prints the ACTG Window's frame but uses a white background (Reverse Video).

The **Exit** option in the File Menu stops execution of the ApoCom **GRAIL™** Interactive Tool (ACTG).

Choose the Save option under the File Menu. This will save the current state of the analysis in the file SAMPLE2.jgr (by default) in the directory containing the original sequence file unless you change the output file name.

Select Print or Save Image if you want to record graphical images of your analysis.

4.6.5.2 Grail 1-1a-2 Algorithm Menu

Clicking on **Grail 1**, **Grail 1a**, or **Grail 2** under the Algorithm Menu changes the display of information related to that option of ApoCom **GRAIL™** analysis. (**Grail 2** is the default.) The differences between the three analysis algorithms are as follows:

Table 2. ApoCom GRAIL™ Analysis Algorithms

Grail 1 recognizes coding potential without using other signals and is best suited for those cases when small fragments are to be evaluated or when genomic context is considered to be inappropriate (as in cDNA sequences).

Grail 1a is an updated version of **Grail 1**. It first uses a fixed-length window to locate the potential coding regions and then evaluates a number of discrete candidates of different lengths around each potential coding region, using information from the two 60-base regions adjacent to that coding region to find the "best" boundaries for each such region. **Grail 1a**, like **Grail 1**, is more useful for non-genomic sequences (such as cDNA and

EST sequences).

Grail 2 uses variable-length windows tailored to each potential exon candidate. It identifies exons by using signals such as splice junctions and other genomic signals. It is best-suited for analysis of genomic sequences.

The initial ApoCom GRAIL™ analysis displayed in the ACTG and ANALYSIS Windows is computed using the Grail 2 analysis option. For the supplied sequence SAMPLE2, forty-one Standard Exons are identified with this analysis option.

Click on the Grail 1 menu option. Notice how the exon potentials change in the ACTG Window. Also, the ANALYSIS Window reflects the different numbers and arrangement of exons. Forty-five exons for SAMPLE2 are found with this analysis option.

Now click on the Grail 1a menu option. Again, the exon potentials will change. This analysis option of ApoCom GRAIL™ identifies 38 Standard Exons within the SAMPLE2 sequence.

Gene Models can be constructed only from Grail 2 exons. For the remainder of the discussion in this chapter, we will be using this analysis option. Click on the Grail 2 menu option.

4.6.5.3 Features Menu

ApoCom GRAIL™ can identify certain DNA features from the given sequence. The Features Menu is used to toggle the graphical display of any of these feature types in the ACTG and DNA SEQUENCE Windows. Specific tabular information about particular feature sites is available through the FEATURES Window.

When the option for a given feature is “selected,” the features are currently being displayed and will be turned OFF if you click on this item. For example, if the “CpG Island” option is not “selected” in this dialog box, the CpG islands will be turned ON after you click on this menu option. If the analysis and identification has not previously been done for a given feature, ApoCom GRAIL™ initiates such an analysis when the feature is selected.

4.6.5.3.1 PolyA Sites

The vertical green bars above and below the GC band of the ACTG Window mark the positions of potential PolyA sites. Within the DNA SEQUENCE Window, corresponding green bars also appear.

Click on the “PolyA Site” option in the Features Menu.

Select a PolyA site marker in the ACTG Window by clicking on the vertical green bar (there are twelve of them in the SAMPLE2 sequence).

Notice that a green bar appears in the DNA SEQUENCE Window to mark the 6-base DNA sequence ('AATAAA' on the Forward strand or 'AAATAA' on the Reverse).

4.6.5.3.2 Promoters

Pol II promoter regions¹⁸ are displayed as solid, pink rectangles above or below the GC band in the ACTG Window. A corresponding pink bar appears in the DNA SEQUENCE Window.

Under the Features Menu, click on the “Promoters” item.

Pol II promoters without a TATA-like element will usually not be found.

Select a pink promoter box in the ACTG Window (there are nine of them within the SAMPLE2 sequence) and the DNA SEQUENCE Window will jump to display the corresponding sequence portion.

Move the DNA SEQUENCE Window across your chosen promoter box. All promoter sites (TATA box, CAAT box, GC box, and cap site) are marked with a pink bar.

¹⁸ Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. *Basic local alignment search tool*. J Mol Biol 1990; 215(3): 403-410.

4.6.5.3.3 CpG Islands

CpG islands are displayed as hollow, magenta rectangles superimposed over the GC band in the ACTG Window. Similar rectangles appear in the DNA SEQUENCE Window.

Under the Features Menu, click on the "CpG Island" menu option.

Select a magenta box in the ACTG Window (there are six of them) and the DNA SEQUENCE Window will jump to display a sequence portion included within the box.

4.6.5.3.4 Repetitive DNA Elements

An option for locating various repetitive DNA elements is provided. These elements are indicated by solid red rectangles.

Under the Features Menu, click on the "Reptiv DNA" item. A (lengthy) analysis will be performed.

Analysis of repetitives in DNA sequences requires detailed sequence comparison using the Smith-Waterman algorithm and may take several minutes, especially for very long sequences or those with many repetitive elements. The human repetitive DNA annotations come from a library provided by J. Jurka.¹⁹

Select a solid red box in the ACTG Window (there are 27 of them for the SAMPLE2 sequence). The DNA SEQUENCE Window will jump to display corresponding red boxes for this subsequence.

4.6.5.3.5 GC Content

¹⁹ Jurka J, Walichiewicz J, Milosavljevic A. J. Mol. Evol. 1992; 35: 286-291.

The relative amount of GC (guanine+cytosine) is represented by gray shading in a central horizontal band in the ACTG Window. This band reflects the GC content of a sliding, 50-base region with white being high GC and black low GC.

4.6.5.3.6 Shadow Exons

In some cases the neural network will recognize the same exon on both strands of the DNA. Where exon predictions overlap on both strands, **Grail 1a** and **Grail 2** incorporate a strand-determination algorithm to choose the more likely strand and to remove spatially-impossible exon candidates. Rejected exon coding probabilities can be toggled (turned ON or OFF) using this option.

Watch the ACTG Window as you toggle the “Shadow Exons” ON and OFF. You will see the rejected exons displayed and then removed.

Change to Grail 1 and Grail 1a. For each analysis option, again toggle the “Shadow Exons” ON and OFF. You should see no change in the Grail 1 ACTG Window while the changes in the Grail 1a ACTG Window will be similar to those seen under Grail 2.

Before proceeding with the tutorial portion of this chapter of the ApoCom GRAIL™ Personal Edition User's Guide, make sure you have all features displayed except the Shadow Exons. Also, be sure you have selected the exons from the Grail 2 analysis algorithm.

4.6.5.4 Window Menu

Clicking on the Window Menu displays a list of all the different windows that ACTG can display:

- DNA Sequence
- Analysis
- Features

Selecting one of these options will display the corresponding pop-up window (if it has been closed). If the selected window is currently being displayed, that window will be brought to the forefront of your screen.

4.6.5.4.1 FEATURES Window

The Features Menu option of the ACTG WINDOW menu bar can be used to toggle "ON" and "OFF" the display of features in the ACTG and DNA SEQUENCE Windows. Any features you wish to display with the FEATURES Window must first be analyzed by the ApoCom GRAIL™. Thus, it is best to select all desired features from the Features Menu before accessing the FEATURES Window.

Select the Features option from the Windows menu.

The FEATURES Window displays tabular information about the list of features found in the sequence by ApoCom GRAIL™ (polyadenylation sites, promoters, CpG islands, and repetitive DNA elements). The table of information to be displayed is selected from a set of options found on the left-hand side of the window. Individual feature items are selected by clicking on their entry in the list. Items can also be selected by clicking on their graphical representation in the ACTG Window. The selected item's information line will be highlighted within the FEATURES Window.

If the number of instances of a given feature within the given DNA sequence is more than the number of items that can be displayed on your screen, you may use the Scrollbar on the right-hand side of the window to scroll through the list.

The **File Menu** item of the FEATURES Window is different than that of the ACTG Window. The **Print** option does not print the image in the FEATURES Window as it appears on the screen, but all of the important information is printed in tabular form. The **Save Image** option allows you to save a GIF image from the FEATURES Window in only two distinct forms:

- **Default:** As you see the image on the screen, but without the Feature Window's frame.
- **Print Frame:** The Feature Window's frame is saved.

The FEATURES Window also provides a "Cut-and-Paste to Clipboard" feature for easy export of this data to your annotation and technical reports. By selecting the **Copy Menu** item, you may copy the **Entire List** or a single selected **List Entry** to the Clipboard for easy export to other application programs.

Display the PolyA Site Table in the FEATURES Window.

Select the **SAMPLE2** sequence's tenth PolyA Site from the display window by clicking on the appropriate line. This site is contained in the Reverse strand from base 43698 to 43703 and has a score of 0.901. This score is a relative measure of the **GRAIL™** neural network recognition confidence.

The display automatically jumps to the corresponding position of this PolyA site in the **ACTG** and **DNA SEQUENCE** Windows. Notice that each graphic has been outlined in yellow.

Click on the PolyA site bar to the right of the selected one. Notice how the display in the **FEATURES** Window changes to highlight the previous item in the Table of PolyA Sites.

You might want to print this entire table on your printer by selecting the Print option in the File Menu.

Display the Promoters Table in the FEATURES Window.

Select the seventh promoter from the **FEATURES** Window by clicking on that line. This site is contained in the Reverse strand from base 54524 to 54941 and has a score of 0.793. This score is a relative measure of the **GRAIL™** neural network recognition confidence.

Also, within this seventh promoter was identified a **CAAT** box at base 54524, a **TATA** box at 54852, a **Cap** site at 54885, and a translation start (**ATG**) site at 54939. No **GC** box was found (indicated by a -1).

The corresponding position of this promoter site is also displayed in the **ACTG** Window. Notice how the pink box has been outlined in cyan.

Click on the promoter site just to the left of the selected one. Notice how the display in the FEATURES Window changes to highlight a different line (line 8 if you choose the Reverse strand promoter or line 1 if you choose the Forward strand promoter).

You might want to copy this entire table to your Clipboard and then into another application by selecting the Entire List option in the Copy Menu.

Display the CpG Island Table in the FEATURES Window.

Select the first line from the display in the FEATURES Window. This site ranges from base 2543 to 2768 (using Forward strand numbering). It has a score of 0.81 and GC content of 51.63%.

The corresponding position of this CpG Island is now shown in the ACTG Window. Notice how the selected CpG Island graphic marker is now outlined in yellow.

Move the display in the ACTG Window down the sequence to find the next two CpG Island markers. Click on one of them. This will outline the graphic in yellow in the ACTG Window and highlight the corresponding line in the FEATURES Window display.

You might want to save the image of this table in GIF format by selecting the Save Image option in the File Menu (choosing to include the Feature Window's frame).

Check to see that the Repttv DNA feature is selected in the Features Menu in the ACTG Window.

Display the Reptv DNA Table in the FEATURES Window.

Select the tenth repetitive DNA item from the FEATURES Window display by clicking on that line. This site is an Alu repetitive element oriented on the Reverse strand from base 28903 to 29200 (using Forward Strand numbering) and has a score of 241.668. This score is computed by the Smith-Waterman algorithm to measure the alignment of repetitive DNA elements to the input sequence.

The corresponding position of this repetitive DNA site is now shown in the ACTG Window. Notice how the graphic is now outlined in cyan.

Locate the repetitive DNA site to the right of the selected one. Click on this new site. This will select this site and highlight a different line in the FEATURES Window display.

You might want to copy the information about the Alu repetitive element oriented on the Reverse strand from base 28903 to 29200 from this table to your Clipboard and then to another application by highlighting the entry and then selecting the List Entry option in the Copy Menu.

Minimize or Close the FEATURES Window for the remaining Tutorial Steps.

4.6.5.5 Assemble Menu

*NOTE: Before using the Assemble option, check to see that the **Grail 2** option has been selected in the Algorithm Menu.*

This menu is used to construct gene models within specified regions of the sequence. The region for assembly is defined by typing in the range in the supplied **Range Boxes**. This version of the Gene Assembly Program (GAP III) uses dynamic programming and heuristics and takes a few minutes to complete. GAP III looks across the entire Forward and Reverse Strands for a possible Gene Model and may predict exons that were not

predicted by the Grail 2 algorithm. The current version of ApoCom GRAIL™ assembles exons on both the Forward strand and Reverse Strand. A particular gene model can be selected by clicking on the corresponding row of the Gene Model Table. Then the results of model construction can be viewed in the ACTG Window as a series of linked yellow bars and in the DNA SEQUENCE Window as yellow bars. The details of the model are listed in the Model Exon Table and Gene Model Table.

Click on the Assemble Menu option (for assembly over the entire set of Grail 2 exons).

After a few minutes, the GENE MODEL TABLE within the ANALYSIS Window will contain entries. This Gene Model will span almost the entire DNA sequence of the Forward Strand.

In the ANALYSIS Window, switch to the GENE MODEL TABLE and select the model on the Forward Strand (containing 27 exons).

Notice the new linked yellow bars displayed above potential exon markers in the ACTG Window and the additional yellow bars in the DNA SEQUENCE Window over portions of the input sequence that are included within the Gene Model.

Compare the exons in the MODEL EXON TABLE with those in the GRAIL™ EXON TABLE. All but one (positioned from base pairs 21111 to 21215) of the Model Exons will be associated with the previously-found exons in the Forward Strand.

Select the model on the Reverse Strand and move to the MODEL EXONS TABLE in the ANALYSIS Window.

ApoCom GRAIL™ will look across the entire Forward and Reverse strands for a possible Gene Model. The model found will include the exon from base 4020 to 4037 and another at 20895 to 21017. The latter exon (found by the Gene Assembly Program) does not have a corresponding entry in the GRAIL™ EXON TABLE.

Select the Model Exon located from 40036 to 40103.

The ACTG Window is now positioned to display the cluster of five exons on the Reverse Strand that are included in the previously-assembled Gene Model.

If the entire set of Grail 2 exons is not to be considered for a Gene Model, the user can restrict the region of consideration by typing the range in the supplied Range Boxes.

Enter these values into the "Assemble from/to" Range Boxes: [17866,27045].

Again, click on the Assemble Menu option.

A particular exon or model must first be selected for a number of operations. As was done for exons in the GRAIL™ Exon Table, selection of Model Exon or Gene Model entries is done by clicking on the desired exon bar in the ACTG or DNA SEQUENCE Window or by clicking on the corresponding row in the Model Exon or Gene Model Table.

When there is more than one Gene Model, only the Model Exons from the currently-selected Gene Model will be displayed in the Model Exon Table. To view the component exons or see the linked bars from another Gene Model, click on the desired model's entry in the Gene Model Table.

4.6.5.6 Zoom Slider and Scrollbar

Other portions of the ApoCom GRAIL™ analysis can be seen by using the Scrollbar (faster) or Arrows (slower) located at the bottom of the ACTG Window. The initial width of the ACTG Window depends upon your display (usually about 5kb wide). Longer sequences can be fit into the window by using the Zoom features.

The Zoom Slider allows for rescaling of the loaded sequence in the ACTG Window. The default zoom value is "x9" and corresponds to about 5 kb per screen width (for most displays). The zoom can be changed (from "x1," "x2," ..., "x9") by dragging the Zoom Slider.

The numerical readout for SAMPLE2 in the Zoom Slider will be "x9" by default.

Drag the Zoom Slider until the numerical readout is "x5."

4.6.5.7 Exit Option

This option found under the File Menu ends execution of the **ApoCom GRAIL™ Personal Edition**. If you select this option, you will first be prompted to save the current state of the analysis in a file. This file can have any extension (“*.jgr” by default) and can be saved in "Complementary Mode." Selecting “Complementary Mode” from this dialog box allows you to save regions on the reverse strand as the complement of regions on the forward strand.

5.0 Summary and Conclusions

ApoCom has been very successful in developing seven products during the Phase II grant period. We are now well poised to expand our current customer base significantly during the Phase III commercialization effort. We are currently seeking outside investment capital to ensure that we are successful in the Phase III commercialization endeavor.

- ApoCom Client Tool for Genomics (ACTG)TM--a multi-platform client/server JavaTM graphical user interface which displays many regions of interest including pol II promoters, repetitives, CpG eslands, polyA sites, protein translations and gene models.
- ApoCom GRAILTM--a commercial server version of the Department of Energy developed GRAILTM Gene Characterization Software licensed to ApoCom by Lockheed Martin Energy Research Corporation (LMER) in 1994. GRAIL is a trademark of LMER.
- ApoCom GRAILTM Network Edition—a multi-platform client/server edition which allows the combination of ALL resources (UNIX®, Windows® 95 and Windows NT®, and MacTM OS) for genomic, cDNA, and EST sequence exploration and discovery.
- ApoCom GRAILTM Personal Edition—a single user, single PC JavaTM edition which unleashes the power of the desktop to analyze sequences both interactively and by automated computation.
- ApoCom GRAILTM Batch Tool—a Visual C++ batch interface for the personal PC for initiating the automated analysis of over 30,000 sequences with a single request.
- ApoCom GRAILTM Command Line Client for Network Edition—a sockets-based command-line interface for easy integration into existing bioinformatics analysis pipelines.
- BATCH ApoCom GRAILTM for Network Edition—a client/server implementation for automated sequence analysis.

BIBLIOGRAPHY

Abergel C, Claverie J-M. *A strong propensity toward loop formation characterizes the expressed reading frames of the D segments at the Ig H and T cell receptor loci.* Eur J Immunol 1991;21(12):3021-3025.

Abola EE, Berstein FC, Bryant SH, Koetzle TF, Weng J. *Protein data bank.* In "Crystallographic Databases-Information Content, Software Systems, Scientific Applications," F. H. Allen, G. Begerhoff and R. Sievers, eds. Data Commission of the International Union of Crystallography, Cambridge; 1987:107-132.

Abremski K, Sirotkin K, Lapedes A. *Application of neural networks and information theory to the identification of E. coli transcriptional promoters.* In: Avula XJR, ed. Mathematical Modelling and Scientific Computing, Vol 2. St. Louis: Principia Scientia, 1993;636-641.

Akrigg D, Attwood TK, Bleasby AJ, Findlay JB, North AC, Maughan NA, Parry-Smith DJ, Perkins DN, Wootton JC. *SERPENT--an information storage and analysis resource for protein sequences.* Comput Appl Biosci 1992;8(3):295-296.

Alpern B., Carter L, Gatlin KS. *Microparallelism and High-Performance Protein Matching.* Association for Computing Machinery, Inc. 1995.

Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.* Nucleic Acids Res. 1997;25:3389-402.

Altschul SF, Boguski MS, Gish W, Wootton JC. *Issues in Searching Molecular Sequence Databases.* Nature Genetics 1994;6:119-129.

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. *Basic Local Alignment Search Tool.* J Mol Biol 1990;215:403-410.

Altschul SF. *Gap costs for multiple sequence alignment.* J Theor Biol 1989;138:297-309.

Altschul SF. *Leaf pairs and tree dissections.* SIAM J Discrete Math 1989;2:293-299.

Altschul SF, Carroll RJ, Lipman DJ. *Weights for data related by a tree.* J Mol Biol 1989;207:647-653.

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. *Basic local alignment search tool.* J Mol Biol 1990;215(3):403-410.

Altschul SF, Lipman DJ. *Protein database searches for multiple alignments.* Proc Natl Acad Sci USA 1990;87(14):5509-5513.

Altschul SF. *Amino acid substitution matrices from an information theoretic perspective.* J Mol Biol 1991;219:555-565.

Altschul SF. *A protein alignment scoring system sensitive at all evolutionary distances.* J Mol Evol 1993;36:290-300.

Altschul SF, Boguski MS, Gish W, Wootton JC. *Issues in searching molecular sequence databases [Review].* Nature Genetics 1994;6:119-129.

Anderson DJ, Fleischmann, ED, and Fry JH. *High Performance Protein Sequence Comparison on Cray Research Computers.* Technical Report September 20, 1991.

Anderson SGE, Kurland CG. *Codon preferences in free living microorganisms.* Microbiol Rev 1990;54:198-210.

Ansari-Lari MA, Muzny DM, Lu J, Lu F, Lilley CD, Spanos S, Malley T, Gibbs RA. *A gene-rich cluster between the CD4 and triosephosphate isomerase genes at human chromosome 12p13.* Genome Res. 1996;6:314-326.

Bairoch A, Boeckmann B. Nucl. Acids Res.1992, 20: 2019-2022.

Bairoch A. Nucl. Acids Res.1993, 21: 3097-3103.

Bairoch A, Bucher P. *PROSITE: Recent Developments.* Nucleic Acids Research 1994;22: 3583-3589.

Bairoch A, Boeckmann B. *SWISS-PROT Protein Sequence Data Bank: Current Status.* Nucleic Acids Research 1994;22:3578-3580.

Baldi P, Brunak S, Chauvin Y, Engelbrecht J, Krogh A. *Periodic sequence patterns in human exons.* ISMB95 1995:30-38.

Barrick D, Villanueva K, Childs J, Kalil R, Schneider TD, Lawrence CE, Gold L, Stormo GD. *Quantitative analysis of ribosome binding sites in E. coli.* Nucleic Acids Research 1994;22:1287-1295.

Baxevas AD, Landsman D. *Histone sequence database: A compilation of highly-conserved nucleoprotein sequences.* Nucleic Acids Research(in press).

Baxevas AD, Boguski MS, Ouellette BFF. *Computational analysis of DNA and protein sequences.* In: Hieter P, ed. Genome Analysis: A Laboratory Manual. Cold Spring Harbor Laboratory, (in press).

Baxevas A, Bryant S, Landsman D. *Homology model building of the HMG-1 box structural domain.* Nucleic Acids Research1995;23(6):1019-1029.

Baxevanis AD, Landsman D. *The HMG-1 box protein family: Classification and functional relationships*. Nucleic Acids Research 1995;23(9):1604-1613.

Baxevanis AD, Arents G, Moudrianakis EN, Landsman D. *A variety of DNA-binding and multimeric proteins contain the histone fold motif*. Nucleic Acids Research 1995;23(14):2685-2691.

Benson D, Boguski MS, Lipman DJ, Ostell J. *The National Center for Biotechnology Information*. Genomics 1990;6:389-391.

Benson D, Lipman DJ, Ostell J. *GenBank*. Nucleic Acids Research 1993;21(13):2963-2965.

Benson DA, Boguski M, Lipman DJ, Ostell J. *GenBank*. Nucleic Acids Research 1994;22:3441-3444.

Bernardi G. *The vertebrate genome: isochores and evolution*. Mol Biol Evol 1993;10:186-204.

Bernardi G. *The isochore organization of the human genome*. Annu Rev Genet 1989;23:637-661.

Beutler E, Gelbart T, Han J, Koziol JA, Beutler B. *Evolution of the genome and the genetic code: selection at the dinucleotide level by methylation and polyribonucleotide cleavage*. Proc Natl Acad Sci USA 1989;86:192-196.

Black DL. *Does steric interference between splice sites block the splicing of a short c-src neuron-specific exon in non-neuronal cells?* Genes Dev 1991;5:389-402.

Blaisdell BE, Campbell AM, and Karlin S. *Genomic comparisons among phages*. Proc Natl Acad Sci USA 1996;93:5854-5859.

Blaisdell BE, Rudd KE, Matin A, Karlin S. *Significant dispersed recurrent DNA sequences in the Escherichia coli genome: several new groups*. J Mol Biol 1993;229:833-848.

Bleasby AJ, Wootton JC. *Construction of validated, non-redundant composite protein sequence databases*. Protein Eng 1990;3:153-159.

Boguski MS. *A molecular biologist visits Jurassic Park*. Biotechniques 1992;12:668-9.

Boguski MS, Hardison RC, Schwartz S, Miller W. *Analysis of conserved domains and sequence motifs in cellular regulatory proteins and locus control regions using new software tools for multiple alignment and visualization*. New Biol 1992;4:247-260.

- Boguski MS. *Computational sequence analysis revisited: New databases, software tools and the research opportunities they engender*. J Lipid Res 1992;33:957-974.
- Boguski M, Claverie J-M, Tolstoshev C, Gish W, Kerlavage A, Adams M, Fields C. *A public resource for expressed sequence tags*. Genome Mapping and Sequencing. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1992:Abstract #225.
- Boguski MS. *A molecular biologist visits Jurassic Park*. Biotechniques 1992;12(5):668-669.
- Boguski MS, Murray AW, Powers S. *Novel repetitive sequence motifs in the alpha and beta subunits of prenyl-protein transferases and homology of the alpha subunit to the MAD2 gene product of yeast [letter]*. New Biol 1992;4(4):408-411.
- Boguski MS, Bairoch A, Attwood TK, Michaels GS. *Proto-vav and gene expression [letter]*. Nature 1992;358(6382):103.
- Boguski MS, States DJ. *Molecular sequence databases and their uses*. In: Rees AR, Wetzel R, Sternberg MJE, eds. Protein Engineering: a Practical Approach. Oxford: IRL Press, 1992;57-88.
- Boguski MS, Lowe TMJ, Tolstoshev CM. *dbEST--database for "expressed sequence tags"*. Nature Genet 1993;4:332-333.
- Boguski MS. *Bioinformatics*. Curr Opin Genet Dev 1994;4:383-388.
- Boguski MS, Tolstoshev CM, Bassett D Jr. *Gene discovery in dbEST*. Science 1994;265:1993-1994.
- Boguski MS. *Adventures in information space: biomedical discoveries in a molecular sequence milieu*. Serials Libr 1995;25(3/4):125-131.
- Boguski MS, Schuler GD. *ESTablishing a human transcript map*. Nat Genet 1995;10:369-371.
- Boguski MS. *The turning point in genome research*. Trends Biochem Sci 1995;20:295-296.
- Boguski MS. *Hunting for genes in computer databases*. New Engl J Med 1995; 333(10):645-647.
- Bork P, Hoffman K, Bucher P, Neuwald AF, Altschul SF, Koonin EV. *A superfamily of conserved domains in DNA damage-responsive cell cycle checkpoint proteins*. FASEB J 1997;11:68-76.

Bork P, Koonin EV. *A P-loop-like motif in a widespread ATP pyrophosphatase domain--implications for the evolution of sequence motifs and enzyme activity*. Proteins 1994; 20: 347-355.

Borodovsky M, McIninch J, Koonin EV, Rudd KE, Medigue C, Danchin A. *Detection of new genes in a bacterial genome using Markov models for three gene classes*. Nucleic Acids Research 1995;23:3554-3562.

Borodovsky M, Rudd KE, Koonin EV. *Intrinsic and extrinsic approaches for detecting genes in a bacterial genome*. Nucleic Acids Research 1994;22:4756-4767.

Borodovsky M, Peresetsky A. *Deriving non-homogeneous DNA Markov chain models by cluster analysis algorithm minimizing multiple alignment entropy*. Computer & Chemistry 1994;18(3):259-268.

Borodovsky M, Koonin EV, Rudd KE. *New genes in old sequence: a strategy for finding genes in the bacterial genome*. Trends in Biochemical Sciences 1994;19:309-313.

Borodovsky M, McIninch J. *GENMARK: parallel gene recognition for both DNA strands*. Comp Chem 1993;17:123-133.

Bouffard G, Ostell J, Rudd KE. *GeneScope: a relational database of Escherichia coli genomic map data for Macintosh computers*. Comput Appl Biosci 1992;8(6):563-567.

Brendel V. *PROSET-A fast procedure to create non-redundant sets of protein sequences*. Math Comput Modelling 1992;16:37-43.

Brendel V, Trifonov EN. *A Computer Algorithm for Testing Prokaryotic Terminators*. Nucleic Acids Research 1984;12:4411-4427.

Breslauer KJ, Frank R, Blocker H, Marky LA. *Predicting DNA duplex stability from the base sequence*. Proc Natl Acad Sci USA 1986;83:3746-3750.

Bucher P. *Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences*. J Mol Biol 1990;212:563-578.

Burge C. *Identification of complete gene structures in human genomic DNA*. PhD thesis. Stanford University, Stanford, CA.

Burge C, Karlin S. *Prediction of Complete Gene Structures in Human Genomic DNA*. J Mol Biol 1997;268:1-17.

Burks C, Cinkosky MJ, Fisher WM, Gilna P, Hayden JE-D, Keen GM, Kelly M, Kristofferson D, Lawrence J. *GenBank*. Nucleic Acids Res. 1992;20:2065-2069.

Burset M, Guigo, R. *Evaluation of Gene Structure Prediction Programs*. Genomics 1996; 34(3): 353-367.

Califano A, Rigoutsos I. In: "Proceeding of the First International Conference on Intelligent Systems for Molecular Biology," July, 1993, Bethesda, MD.

Calladine CR, Drew HR. *Understanding DNA*. 1992, Academic Press, San Diego.

Chen IT, Chasin LA. *Large exon size does not limit splicing in vivo*. Mol Cell Biol 1994;14:2140-2146.

Chou PY, Fasman GD. *Prediction of the Secondary Structure of Proteins From Their Amino Acid Sequence*. Advances in Enzymology 1978;47:45-147.

Church DM, Stotler CJ, Rutter JL, Murrell JR, Trofatter JA, Buckler AJ. *Isolation of genes from complex sources of mammalian genomic DNA using exon amplification*. Nature Genetics 1994;6:98-105.

Claverie J-M. *Du traitement de l'information a l'evaluation des strategies genome research*. Biofutur 1990;94:22-27.

Claverie J-M, Sauvaget I. *WOBB.C: a portable software package for defining and searching ambiguous sequence patterns*. Protein Seq Data Anal 1991;4(2):119-121.

Claverie J-M, Daulmerie C. *Smoothing profiles with sliding windows: better to wear a hat!* Comput Appl Biosci 1991;7(1):113-115.

Claverie J-M. *Identifying coding exons by similarity search: Alu-derived and other potentially misleading protein sequences*. Genomics 1992;12:838-841.

Claverie J-M, States D, Gish W. *Computer methods for exon identification in human genomic sequences: case studies and critical assessment*. Genome Mapping and Sequencing. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1992;Abstract #224.

Claverie J-M. *Sequence signals: artifact or reality?* Comput Chem 1992;16:89.

Claverie J-M. *L'ordinateur a la chasse au genes*. Sci Vie 1992;181:104-113.

Claverie J-M, Hardelin JP, Legouis R, Levilliers J, Bougueleret L, Mattei MG, Petit C. *Characterization and chromosomal assignment of a human cDNA encoding a protein related to the murine 102-kDa cadherin-associated protein*. Genomics 1993;15(1):13-20.

Claverie J-M, States D. *Information enhancement methods for large scale sequence analysis*. Comput Chem 1993;17:191-201.

- Claverie J-M. *A database of ancient protein sequences [letter]*. Nature 1993;364:19-20.
- Claverie J-M. *Large Scale Sequence Analysis*. In: Automated Sequencing and Analysis. (Adams MD, Fields C, Venter JC, eds.) New York: Academic Press, 1994.
- Claverie J-M, Makalowski W. *Alu alert (letter)*. Nature 1994;371:752.
- Claverie J-M. *Detecting frameshifts by amino acid comparison*. J Mol Biol 1994; 234(4): 1140-1157.
- Claverie J-M. *Shallow shotgun sequencing as a strategy for finding coding exons*. In: Hochgeschwender U, Gardiner K, eds. Identification of Transcribed Sequences. New York: Plenum Press, 1994;229-237.
- Claverie J-M. *A streamlined random sequencing strategy for finding coding exons*. Genomics 1994;23:575-581.
- Claverie J-M. *Some useful statistical properties of position-weight matrices*. Computer & Chemistry 1994;18(3):287-294.
- Dayhoff MO, Schwartz RM, Orcutt BC. 1978. In "Atlas of Protein Sequences and Structure," (Dayhoff, M. O. ed). Nat. Biomed. Res. Found., Washington, D. C. 1978; 5 (Suppl. 3):345-352.
- Dominski Z, Kole R. *Selection of splice sites in pre-mRNAs with short internal exons*. Mol Cell Biol 1991;11:6075-6083.
- Dong S, Searls DB. *Gene structure prediction by linguistic methods*. Genomics 1994;23:540-551.
- Doniger J, Landsman D, Gonda MA, Wistow G. *The product of unr, the highly conserved gene upstream of N-ras, contains multiple repeats similar to the cold-shock domain (CSD), a putative DNA-binding motif*. New Biol 1992;4(4):389-395.
- Duret L, Mouchiroud D, Gautier C. *Statistical analysis of vertebrate sequences reveals that long genes are scarce in CG-rich isochores*. J Mol Evol 1995;40:308-317.
- Emmert DB, Stoehr PJ, Stoesser G, Cameron GN. *The European Bioinformatics Institute (EBI) Databases*. Nucleic Acids Research 1994;22:3445-3449.
- Engelman DM, Steitz TA, Goldman A. *Identifying Nonpolar Transbilayer Helices in Amino Acid Sequences of Membrane Proteins*. Annual Review of Biophysics and Biophysical Chemistry 1986;15:321-353.

Farber R, Lapedes A, Sirotkin K. *Determination of eukaryotic protein coding regions using neural networks and information theory.* J Mol Biol 1992;226:471-479.

Feng DF, Doolittle RF. *Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees.* Journal of Molecular Evolution 1987;25:351-360.

Fetrow JS, Bryant SH. *New programs for protein tertiary structure prediction.* Biotechnology 1993;11:479-483.

Fickett JW. *Recognition of Protein Coding Regions in DNA Sequences.* Nucleic Acids Research 1982;10:5303-5318.

Fickett JW, Torney DC, Wolf DR. *Base compositional structure of genomes.* Genomics 1992;13:1056-1064.

Fickett JW, Tung C-S. *Assessment of protein coding measures.* Nucl Acids Res 1992;20:6441-6450.

Fickett JW. *Inferring genes from open reading frames.* Computer & Chemistry 1994;18(3):203-205.

Fickett JW. *The gene identification problem: an overview for developers.* Computer & Chemistry 1995;20:103-118.

Fickett JW. *ORFs and genes: How strong a connection?* Journal of Computational Biology 1995;2(1):117-123.

Fickett JW, Guigo R. *Computational gene identification.* In: Internet for the Molecular Biologist (Swindell S, ed.) 1995.

Fickett JW. *Finding Genes By Computer: The State Of The Art.* Trends in Genetics 1996;12(8):316-320.

Fields C. *Integrating computational and experimental methods for gene discovery.* In: Automated Sequencing and Analysis. (Adams MD, Fields C, Venter JC, eds.) New York: Academic Press, 1994;321-325.

Filipski J. In: Advances in Mutagenesis Research (Obe, G, ed.) Vol 2, Springer-Verlag, Heilelberg, Germany, 1990:1-54.

Fleshman J, Rapp B, Benson D. *GenBank at NIH: Database grows by 1,000 sequences per week.* NIH Catalyst 1994;March:8-9.

Forney GD. *The Viterbi algorithm.* Proc IEEE 1973;61:268-278.

- Garnier J, Osguthorpe DJ, Robson B. *Analysis of the Accuracy and Implications of Simple Methods for Predicting the Secondary Structure of Globular Proteins*. Journal of Molecular Biology 1978;120:97-120.
- Gautheret D, Major F, Cedergren R. *Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA*. Comput Appl Biosci 1990; 6(4): 325-31.
- Gautheret D, Major F, Cedergren R. *Computer modeling and display fo RNA secondary and tertiary structures*. Methods Enzymol 1990;198:318-330.
- Gautheret D, Major F, Cedergren R. *Modeling the three-dimensional structure of RNA using discrete nucleotide conformational sets*. J Mol Biol 1993;229:1049-1064.
- Gelfand MS, Roytberg MA. *Prediction of the intron-exon structure by a dynamic programming approach*. BioSystems 1993;30:173-182.
- Gelfand MS *Prediction of function in DNA sequence analysis*. Journal of Computational Biology 1995;2(1):87-117.
- Gelfand MS, Mironov AA, Pevzner PA. *Gene recognition via spliced sequence alignment*. Proceedings of the National Academy of Sciences USA 1996;93(17):9061-9066.
- Gelfand MS, Podolsky LI, Astakhova TV, Roytberg MA. *Recognition of Genes in Human DNA Sequences*. Journal of Computational Biology 1996;3(2):223-234.
- Geigerich G, Meyer F, Schleiermacher C. *GeneFisher - Software support for the detection of postulated genes*. ISMB96 1996:68-77.
- George DB, Barker WC, Mewes H, Pfeiffer F, Tsugita, A. *The PIR-International Protein Sequence Database*. Nucleic Acid Research 1994;22:3569-3573.
- Ghosh D. *A relational database of transcription factors*. Nucleic Acids Research 1990; 18(7): 1749-1756.
- Ghosh D, Kieff E. *Cis-acting regulatory elements near the Epstein-Barrvirus latent-infection membrane protein transcriptional start site*. J Virol 1990;64(4):1855-1858.
- Ghosh D. *New developments of a transcription factors database*. Trends Biochem Sci 1991;16:445-447.
- Ghosh D. *TFD: the transcription factors database*. Nucleic Acids Research 1992; 20(Suppl): 2091-2093.

Ghosh D. *Status of the transcription factors database (TFD)*. Nucleic Acids Research 1993; 21(3):3117-3118.

Gish W, States DJ. *Identification of protein coding regions by database similarity search*. Nature Genet 1993;3:266-272.

Goldstein L, Waterman MS. *Approximations to profile score distributions*. Journal of Computational Biology 1994;1:93-104.

Gorbalenya AE, Koonin EV. *Helicases: amino acid sequence comparisons and structure-function relationships*. Curr Opin Struct Biol 1993;3:419-3429.

Gotoh O. *An Improved Algorithm for Matching Biological Sequences*. J. Mol. Biol. 1982; 162:705-708.

Green P, Lipman D, Hillier L, Waterston R, States D, Clayerie J-M. *Ancient conserved regions in new gene sequences and the protein databases*. Science 1993;259:1711-1716.

Green P. *Ancient conserved regions in gene sequences*. Current Opinions in Structural Biology 1994;4:404-412.

Gribskov M, Devereux J, Burgess RR. *The Codon Preference Plot: Graphic Analysis of Protein Coding Sequences and Prediction of Gene Expression*. Nucleic Acids Research 1984;12:539-549.

Gribskov M, McLachlan M, Eisenberg D. *Profile Analysis: Detection of Distantly Related Proteins*. Proceedings of the National Academy of Sciences USA 1987;84:4355-4358.

Gribskov M, Homyak, M., Edenfield, J., and Eisenberg, D. *Profile Scanning for Three-Dimensional Structural Patterns in Protein Sequences*. Computer Applications in the Biosciences 1988;4:61-66.

Gribskov M, Luethy R, Eisenberg D. *Profile Analysis*. In: Methods in Enzymology, (Doolittle RF, ed.), Academic Press, San Diego, California, USA, 1990;183:146-159.

Gribskov M, Devereux J. *Sequence Analysis Primer*. Stockton Press, New York, USA, 1991.

Griffin AM, Griffin HG., editors. *Computer Analysis of Sequence Data*, Part 1, Vol. 24 of Methods in Molecular Biology. Humana Press, Inc., Totowa, New Jersey, 1994.

Grosse I, Herzel J, Buldyrev SV, Stanley HE. *Mutual information of coding and noncoding DNA*. Submitted.

Guan X, Uberbacher EC. *Alignment of DNA and Protein Sequences Containing Frameshift Errors*. ORNL/TM-12976, April 1995.

Guigo R, Knudsen S, Drake N, Smith T. *Prediction of gene structure*. J Mol Biol 1992;226:141-157.

Guigo R, Fickett JW. *Distinctive sequence features in protein coding, non-coding, and intergenic human DNA*. Journal of Molecular Biology 1995;253:51-60.

Gutman GA, Hatfield GW. *Nonrandom utilization of codon pairs in Escherichia coli*. Proc Natl Acad Sci USA 1989;86:3699-3703.

Hackathorn RD. *Enterprise Database Connectivity*. Wiley, 1993.

Haggard-Ljungquist E, Jacobsen E, Rishovd S, Six EW, Nilssen O, Sunshine M, Lindquist BH, Kim KJ, Barreiro V, Koonin EV, Calendar R. *Bacteriophage P2: genes involved in baseplate assembly*. Virology (in press).

Hannenhalli S, Chappey C, Koonin EV, Pevzner PA. *Genome sequence comparison and scenarios for genome rearrangements: a test case*. Genomics (in press).

Harris NL, Senepathy P. *Distribution and consensus of branch point signals in eukaryotic genes: a computerized statistical analysis*. Nucl Acids Res 1990;18:3015-3019.

Hawkins JD. *A survey on intron and exon lengths*. Nucl Acids Res 1988;16:9893-9908/

Hebsgaard SM, Korning PG, Tolstrup N, Engelbrecht J, Rouzé P, Brunak S. *Splice site prediction in Arabidopsis thaliana pre-mRNA by combining local and global sequence information*. Nucleic Acids Research 1996;24(17):3439-3452.

Hendrickson W, Rudd KE. *Physical map location of the argFGH operon of Escherichia coli*. J Bacteriol 1992;174(11):3836-3837.

Henikoff S. *Comparative sequence analysis: finding genes*. In: Biocomputing: Informatics and Genome Projects. (Smith DW, ed.) Academic Press, 1994:87-117.

Henikoff S, Henikoff JG. Proc. Natl. Acad. Sci. USA 1992;89:10915-11009.

Hanai R, Wada A. *The effects of guanine and cytosine variation on dinucleotide frequency and amino acid composition in the human genome*. J Mol Evol 1988;27:321-325.

Hanawalt P. *Transcription coupled repair and human disease*. Science 1994;266:1957-1958.

Holmquist GP, Filipski J. *Organization of mutations along the genome: a prime determinant of genome evolution*. Trends Ecol Evol 1994;9:65-69.

Hudson-Taylor DE, Dolan SA, Klotz FW, Fujioka H, Aikawa M, Koonin EV, Miller LH. *Plasmodium falciparum protein associated with the invasion junction contains a conserved oxidoreductase domain*. Mol Microbiol 1995;15:463-471.

Hunter CA. *Sequence-dependent DNA structure. The role of base stacking interactions*. J Mol Biol 1993;230:1025-1054.

Hunter LE, States DJ. *Bayesian classification of protein structural elements*. In: Milutinovic V, Shriver B, eds. Hawaiian International Conference on System Sciences. IEEE Computer Society Press, 1990:595-604.

Hunter L, States D. *Applying Bayesian classification to protein structure*. In: Werner R, ed. Seventh Conference on Artificial Intelligence Applications. IEEE Computer Society Press, 1990:10-16.

Hunter L, Harris NL, States DJ. *Efficient classification of massive unsegmented datastreams*. Proceedings of the International Machine Learning Workshop. Aberdeen, Scotland: Morgan Kaufman Associates, 1992:224-232.

Ikemura T. *Codon usage and tRNA content in unicellular and multicellular organisms*. Mol Biol Evol 1985;2:12-34.

Ilyina TV, Koonin EV. *Conserved sequence motifs in the initiator proteins for rolling circle DNA replication encoded by diverse replicons from eubacteria, eucaryotes and archaeobacteria*. Nucleic Acids Research 1992;20(13):3279-3285.

Iris FJ, Bougueleret L, Prieur S, Caterina D, Primas G, Perrot V, Jurka J, Rodriguez-Tome P, Claverie J-M, Dausset J, Cohen D. *Dense Alu clustering and a potential new member of the NF kappa B family within a 90 kilobase HLA class III segment*. Nature Genet 1993;3:137-145.

Irwin, B, Heck JD, Hatfield, GW. *Codon pair utilization biases influence translational elongation step times*. J Biol Chem 1995;270:22801-22806.

Jayakumar A, Rudd KE, Fabiny JM, Barnes EJ. *Localization of the Escherichia coli amtA gene to 95.8 minutes*. J Bacteriol 1991;173(5):1572-3.

Jaeger JA, Turner DH, Zuker M. *Improved Predictions of Secondary Structures for RNA*. Proceedings of the National Academy of Sciences USA 1989;86:7706-7710.

Jiang B, Monroe SS, Koonin EV, Stine SE, Glass RI. *RNA sequence of astrovirus: unique genomic organization and a putative retrovirus-like ribosomal frameshifting*

signal directing syntheses of the viral replicase. Proc Natl Acad Sci USA 1993;90:10539-10543.

Jin T, Rudd KE, Inouye M. *The nlpA lipoprotein gene is located near the selC tRNA gene on the Escherichia coli chromosome.* J Bacteriol 1992;174(11):3822-3823.

Josse J, Kaiser AD, Kornberg A. *Enzymatic synthesis of deoxyribonucleic acid VIII. Frequencies of nearest neighbor base sequences in deoxyribonucleic acid.* J Biol Chem 1961;236:864-875.

Jukes TH, Bhushan V. *Silent nucleotide substitutions and G+C content of some mitochondrial and bacterial genes.* J Mol Evol 1986;24:39-44.

Julich A. *Implementation of BLAST for Parallel Computers.* CABIOS 1995;11(1):3-6.

Jurka J, Klonowski P, Dagman V, Pelton P. *CENSOR-a program for identification and elimination of repetitive elements from DNA sequences.* Comp Chem 1996;20(1):119-122.

Jurka J, Walichiewicz J, Milosavljevic A. J. Mol. Evol. 1992; 35: 286-291.

Kabat EA, Wu TT, Perry HM, Gottesman KS, Foeller C. *Sequences of Proteins of Immunological Interest.* 5th edition. 3 vols. Bethesda, MD: National Institutes of Health. National Center for Biotechnology Information, NIH publication no. 91-3242, 1991;2597.

Kamb A, Wang C, Thomas A, DeHoff BS, Norris FH, Richardson K, Rine J, Skolnick M, Rosteck PR Jr. *Software trapping: a strategy for finding genes in large genomic regions.* Computers and Biomedical Research. To appear.

Kans JA, Spengler SJ, Cole GM, Mortimer RK. *An electrophoretic method for assaying polymerase and nuclease activities.* Methods Mol Cell Biol 1991;2(6):266-272.

Kans JA, Mortimer RK. *Nucleotide sequence of the RAD57 gene of Saccharomyces cerevisiae.* Gene 1991;105:139-140.

Karasev AV, Boyko VP, Gowda M, Nikolaeva OV, Hilf ME, Koonin EV, Niblett CL, Cline KC, Gumpf DJ, Lee RF, Garnsey SM, Dawson WDO. *Complete sequence of the Citrus tristeza virus RNA genome.* Virology 1995;208:511-520.

Karlin S, Mrazek J. *Genome sequences and bacterial phylogeny.* In: Bacterial Genomes: Physical Structure and Analysis (de Bruijn FJ, Lupski J, Weinstock G., eds) Chapman & Hall, New York, 1997.

Karlin S, Mrazek J. *What Drives Codon Choices in Human Genes?* J Mol Biol 1996;262:459-472.

Karlin S. *Assessing Inhomogeneities in Bacterial Long Genomic Sequences*. RECOMB '97, Santa Fe, New Mexico, ACM 1997:164-170.

Karlin S, Altschul SF. *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes*. Proc Natl Acad Sci USA 1990;87:2264-2268.

Karlin S, Bucher P, Brendel V, Altschul SF. *Statistical methods and insights for protein and DNA sequences*. Annu Rev Biophys Chem 1991;20:175-203.

Karlin S, Bucher P. *Correlation analysis of amino acid usage in protein classes*. Proc Natl Acad Sci USA 1992;89:12165-12169.

Karlin S, Altschul SF. *Applications and statistics for multiple high-scoring segments in molecular sequences*. Proc Natl Acad Sci USA 1993;90:5873-5877.

Karlin S, Mocarski EM, Schachtel G. *Molecular evolution of herpes viruses: genomic and protein sequence comparisons*. J Virol 1994;68:1886-1902.

Karlin S, Burge C. *Dinucleotide relative abundance extremes: a genomic signature*. Trends Genet 1995;11:283-290.

Karlin S, Cardon LR. *Computational DNA sequence analysis*. Annu Rev Microbiol 1994;48:619-654.

Karlin S, Ladunga I. *Comparisons of eukaryotic genomic sequences*. Proc Natl Acad Sci USA 1994;91:12837-12841.

Kel OV, Romachenko AG, Kel AE, Naumochkin A, Kolchanov NA. *Structure of data representation in TRRD - database of transcription regulatory regions on eukaryotic genomes*. In: Proceedings of the 28th Annual Hawaii International Conference on System Sciences (HICSS), Wailea, Hawaii, January 4-7, 1995.

Kernighan BW, Plauger PJ. *Software Tools*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1976.

Kinzler KW, Nilbert MC, Su LK, Vogelstein B, Mryan TM, Levy DB, Smith KJ, Preisinger AC, Hedege P, McKechnie D, Finnear R, Markham A, Groffen J, Boguski MS, Altschul SF, Horii A, Ando H, Miyoshi Y, Miki Y, Nishisho I, Nakamura Y. *Identification of FAP locus genes from chromosome 5q21*. Science 1991;253:661-665.

Klaassen VA, Boeshore M, Koonin EV, Tian T, Falk BW. *Genome structure and phylogenetic analysis of lettuce infectious yellows virus, a whitefly-transmitted, bipartite closterovirus*. Virology 1995;208:99-110.

Knueppel R, Dietze P, Lehnberg W, Frech K, Wingender E. *TRANSFAC retrieval program: A network model database of eukaryotic transcription regulating sequences and proteins*. Journal of Computational Biology 1994;1:191-198.

Kokubo T, Gong DW, Wootton JC, Horikoshi M, Roeder RG, Nakatani Y. *Molecular cloning of Drosophila TFIID subunits*. Nature 1994;367:484-487.

Konopka AK. *Sequences and codes: fundamentals of biomolecular cryptography*. In: Biocomputing: Informatics and Genome Projects. (Smith DW, ed.) Academic Press, 1994.

Koonin EV, Woods DF, Bryant PJ. *A Drosophila tumor suppressor gene product and its nematode and mammalian homologs contain a guanylate kinase-related domain with a disrupted ATP-binding site*. Nature Genet (in press).

Koonin EV, Tatusov RL, Rudd KE. *Sequence similarity analysis of Escherichia coli proteins--functional and evolutionary implications*. Proc Natl Acad Sci USA (in press).

Koonin EV, Zhou S, Lucchesi JC. *Chromo domain--sequence conservation and possible role in delivering transcription regulators to chromatin*. Nucleic Acids Research (in press).

Koonin EV, Bork P, Sander C. *Yeast chromosome III: new gene functions*. EMBO Journal 1994;13:492-503.

Koonin EV. *DnaC protein contains a modified ATP-binding motif and belongs to a novel family of ATPases including also DnaA*. Nucleic Acids Research 1992;20(8):1997.

Koonin EV, Gorbalenya AE, Purdy MA, Rozanov MN, Reyes GR, Bradley DW. *Computer-assisted assignment of functional domains in the nonstructural polyprotein of hepatitis E virus: delineation of an additional group of positive-strand RNA plant and animal viruses*. Proc Natl Acad Sci USA 1992;89(17):8259-8263.

Koonin EV, Senkevich TG, Chernos VI. *A family of DNA virus genes that consists of fused portions of unrelated cellular genes*. Trends Biochem Sci 1992;17(6):213-214.

Koonin EV, Senkevich TG. *Vaccinia virus encodes four putative DNA and/or RNA helicases distantly related to each other*. J Gen Virol 1992;73(Pt 4):989-993.

Koonin EV, Senkevich TG. *Evolution of thymidine and thymidylate kinases: the possibility of independent capture of TK genes by different groups of viruses*. Virus Genes 1992;6(2):187-196.

Koonin EV, Gorbalenya AE. *An insect picornavirus may have genome organization similar to that of caliciviruses*. FEBS Lett 1992;297:81-86.

Koonin EV. *Evolution of double-stranded RNA viruses: a case for polyphyletic origin from different groups of positive-stranded RNA viruses*. Semin Virol 1992;3(5):327-339.

Koonin EV, Ilyina TV. *Geminivirus replication proteins are related to prokaryotic plasmid rolling circle DNA replication initiator proteins*. J Gen Virol 1992;73:2763-2766.

Koonin EV. *Comparative analysis of biopolymer sequences: reflections on the validity of the methodology and underlying general principles*. In: Gindikin S, ed. Mathematical Methods of Analysis of Biopolymer Sequences. Providence, RI: American Mathematical Society (DIMACS Series in Discrete Mathematics and Theoretical Computer Science; vol 8) 1992;63-74.

Koonin EV. *Computer-assisted identification of a putative methyltransferase domain in NS5 protein of flaviviruses and lambda 2 protein of reovirus*. J Gen Virol 1993;74:733-740.

Koonin EV, Deutscher MP. *RNase T shares conserved sequence motifs with DNA proofreading exonucleases*. Nucleic Acid Res 1993;21:2521-2522.

Koonin EV. *A common set of conserved motifs in a vast variety of putative nucleic acid-dependent ATPases including MCM proteins involved in the initiation of eukaryotic DNA replication*. Nucleic Acids Research 1993;21:2541-2547.

Koonin EV, Ilyina TV. *Computer-assisted analysis of the relationships between protein and DNA sequences involved in rolling circle DNA replication*. Biosystems 1993;30:241-268.

Koonin EV. *Genome sequencing: a glimpse of the area at the brink of the virus era*. Biosystems 1993;30:269-276.

Koonin EV, Dolja VV. *Evolution and taxonomy of positive-strand RNA viruses: implications of comparative analysis of amino acid sequences*. Crit Rev Biochem Mol Biol 1993;28:375-430.

Koonin EV. *Escherichia coli dinG gene encodes a putative DNA helicase related to a group of eukaryotic helicases including Rad3 protein*. Nucleic Acids Research 1993;21(6):1497.

Koonin EV. *A highly conserved sequence motif defining the family of MutT-related proteins from eubacteria, eukaryotes and viruses*. Nucleic Acid Res 1993;21:4847.

Koonin EV, Rudd KE. *SpoU protein of Escherichia coli belongs to a new family of putative rRNA methyltransferases*. Nucleic Acids Research 1993;21:5519.

Koonin EV, Bork P, Sander C. *Yeast chromosome III: new gene functions*.

EMBO J 1994;13:493-503.

Koonin EV, Rudd KE. *A conserved domain in putative bacterial and bacteriophage transglycosylases*. Trends in Biochem Sci 1994;19:106-107.

Koonin EV. *Conserved sequence pattern in a wide variety of phosphoesterases*. Protein Sci 1994;3:356-358.

Koonin EV, Bork P, Sander C. *A novel RNA-binding motif in omnipotent suppressor of translation termination, ribosomal proteins and a ribosome modification enzyme*. Nucleic Acids Research 1994;22:2166-2167.

Koonin EV, Bork P. *FMN or DNA binding?* Trends Biochem Sci 1994;19:234-235.

Koonin EV, Tatusov RL. *Computer analysis of bacterial dehalogenases defines a large superfamily of hydrolases with diverse specificity*. J Mol Biol 1994;245:125-132.

Koonin EV, Mushegian AR, Tatusov RL, Altschul SF, Bryant SH, Bork P, Valencia A. *Eukaryotic translation elongation factor 1 gamma contains a glutathione transferase domain--study of a diverse, ancient protein superfamily using motif search and structural modeling*. Protein Sci 1994;3:2045-2054.

Koonin EV, Van der Vies SM. *Conserved sequence motifs in bacterial and bacteriophage chaperonins*. Trends Biochem Sci 1995;20:14-15.

Koonin EV. *A protein splice-junction motif in hedgehog family proteins*. Trends Biochem Sci 1995;20:141-142.

Koonin EV. *Multidomain organization of eukaryotic guanine nucleotide exchange translation initiation factor eIF-2B subunits revealed by analysis of conserved sequence motifs*. Protein Sci 1995;4:1608-1617.

Krogh A, Mian IS, Haussler D. *A hidden Markov model that finds genes in E. coli DNA*. Nucleic Acids Research 1994;22:4768-4778 and UCSC Report CRL-93-33.

Krogh A, Brown M, Mian IS, Sjoelander K, Haussler D. *Hidden Markov models in computational biology. Applications to protein modeling*. Journal of Molecular Biology 1994;235:1501-1531.

Kruskal JB. *An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules*. SIAM Review 1983; 25(2):201-237.

Kulaeva OI, Wootton JC, Levine AS, Woodgate R. *Characterization of the umu-complementing operon from R391*. J Bacteriol 1995;177:2737-2743.

Kulp D, Haussler D, Reese MG, Eechman FH. *A generalized hidden Markov model for the recognition of human genes in DNA*. ISMB96 1996:134-142.

Kurland CG. *Major codon preference: theme and variations*. Biochem Soc Trans 1993;21:841-845.

Kyte J, Doolittle RF. (1982). *A Simple Method for Displaying the Hydropathic Character of a Protein*. Journal of Molecular Biology 1982;157:105-132.

La Spada AR, Paulson HL, Fischbeck KH. *Trinucleotide repeat expansion in neurological disease*. Ann Neurol 1994;36:814-822.

Lai CC, Boguski M, Broek D, Powers S. *Influence of guanine nucleotides on complex formation between Ras and CDC25 proteins*. Mol Cell Biol 1993;13(3):1345-1352.

Landsman D, Bustin M. *Assessment of the transcriptional activation potential of the HMG chromosomal proteins*. Mol Cell Biol 1991;11(9):4528-4536.

Larson NI, Engelbrecht J, Brunak S. *Analysis of Eukaryotic Promoter Sequences Reveals a Systematically Occurring CT-signal*. Nucleic Acids Research 1995;23:1223-1230.

Lasser AB, Buskin JN, Lockshon D, Davis RL, Apone S, Hauschka SD, Weintraub H. *MyoD is a sequence-specific DNA binding protein requiring a region of myc homology to bind to the muscle creatine kinase enhancer*. Cell 1989;58:823-831.

Lawrence CE, Bryant SH. *Hydrophobic potentials from statistical analysis of protein structures*. Methods Enzymol 1991;202(20):20-31.

Lawrence CE, Altschul SF, Boguski MS, Liu JS, Neuwald AF, Wootton JC. *Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment*. Science 1993;262:208-214.

Lawrence CE, Altschul SF, Wootton JC, Boguski MS, Neuwald AF, Liu JS. *A Gibbs Sampler for the Detection of Subtle Motifs in Multiple Sequences*. In: Hunter L, ed. 27th Hawaii International Conference on System Sciences. Maui, Hawaii: IEEE Computer Society Press, 1994:245-254.

Legouis R, Hardelin JP, Levilliers J, Claverie J-M, Compain S, Wunderle V, Millasseau P, Le Paslier D, Cohen D, Caterina D, Bougeuleret L, Lutfall G, Weissenbach J, Petit C. *The candidate gene for the X-linked Kallmann syndrome encodes a protein related to adhesion molecules*. Cell 1991;67(2):423-435.

Li P, Davies CJ, North D, Schilling P, Evans GA, Garner HR. *Supercomputing in Genomic Sequencing: Optimization of BLAST and Other Sequence Algorithms for High Speed Parallel Processing*. Scientific Computing and Automation 1997;11:19-24.

- Lightfoot DA, Baron AJ, Cock JM, Wootton JC. *A nitrate reductase gene of the cyanobacterium Synechococcus PCC6301 inferred by heterologous hybridization, cloning and targeted mutagenesis*. *Genetica* 1992;85(2):107-117.
- Lipman DJ, Pearson, WR. *Rapid and Sensitive Protein Similarity Searches*. *Science* 1985; 227:1435-1441.
- Lipman DJ, Altschul SF, Kececioglu JD. *A tool for multiple sequence alignment*. *Proc Natl Acad Sci USA* 1989;86(12):4412-4415.
- Lipman DJ, Wilbur WJ. *Modelling neutral and selective evolution of protein folding*. *Proc R Soc Lond [Biol]* 1991;245(1312):7-11.
- Lombardo MJ, Miller CG, Rudd KE. *Physical mapping of the Escherichia coli pepT and potABCD genes*. *J Bacteriol* 1993;175(23):7745-7746.
- Lonetto MA, Brown KL, Rudd KE, Buttner MJ. *Analysis of the Streptomyces coelicolor sigE gene reveals the existence of a subfamily of eubacterial RNA polymerase sigma factors involved in the regulation of extracytoplasmic functions*. *Proc Natl Acad Sci USA* 1994;91:7573-7577.
- Lopez R, Larsen F, Prydz H. *Evaluation of the exon predictions of the GRAIL software*. *Genomics* 1994;24:133-136.
- Madej T, Boguski MS, Bryant S. *Threading analysis suggests that the obese gene product may be a helical cytokine*. *FEBS Lett* (in press).
- Maizel JV, Lenk RP. *Enhanced Graphic Matrix Analysis of Nucleic Acid and Protein Sequences*. *Proceedings of the National Academy of Sciences USA* 1981;78:7665-7669.
- Majlessi L, Roth C, Benaroch P, Claverie J-M, Kourilsky P, Bordenave G. *Identification of T cell peptides from C gamma 2ab involved in IgG2ab T cell- induced allotypic suppression. Relationship with the MHC haplotype of the suppressor T cell-donor Igha mice*. *J Immunol* 1993;151(3):1263-1272.
- Major F, Lapalme G, Cedergren R. *Domain generating functions for solving constraint satisfaction problems*. *J Funct Programming* 1991;1(1):213-227.
- Major F, Turcotte M, Gautheret D, Lapalme G, Fillion E, Cedergren R. *The combination of symbolic and numerical computation for three-dimensional modeling of RNA*. *Science* 1991;253:1255-1260.
- Major F, Lapalme G. *Derivation d'une structure de controle de retour-arriere*. In: Queinnec C, ed. *Actes des journees JFLA92: Avancees applicatives*. Perros-Guirec, France: BIGRE, 1992:202-219.

Major F, Gautheret D, Cedergren R. *Reproducing the three-dimensional structure of a tRNA molecule from structural constraints*. Proc Natl Acad Sci USA 1993;90:9408-9412.

Major F. *Computer modeling of RNA 3-dimensional structures*. In: Meyers RA, ed. Molecular Biology and Biotechnology: A Comprehensive Desk Reference. New York: VCH Publishers, 1995;847-850.

Mannella CA, Neuwald AF, Lawrence CE. *Identification of likely transmembrane beta-strand regions in sequences of mitochondrial pore proteins using the Gibbs sampler*. J Bioenerg Biomembr (in press).

Marchuk DA, Saulino AM, Tavakkol R, Swaroop M, Wallace MR, Andersen LB, Mitchell AL, Gutmann DH, Boguski MS, Collins FS. *cDNA cloning of the Type 1 Neurofibromatosis Gene: complete sequence of the NF1 gene product*. Genomics 1991;11:931-940.

Maschat F, Dubertret ML, Therond P, Claverie J-M, Lepesant JA. *Structure of the ecdysone-inducible P1 gene of Drosophila melanogaster*. J Mol Biol 1990;214(2):359-372.

McKeown M. *The role of small nuclear RNAs in RNA splicing*. Curr Opin Cell Biol 1993;5:448-454.

Medigue C, Rouxel T, Vigier P, Henaut A, Danchin A. *Evidence for horizontal gene transfer in Escherichia coli speciation*. J Mol Biol 1991;222:851-856.

Miller W, Ostell J, Rudd KE. *An algorithm for searching restriction maps*. Comput Appl Biosci 1990;6(3):247-252.

Miller W, Barr J, Rudd KE. *Improved algorithms for searching restriction maps*. Comput Appl Biosci 1991;7(4):447-456.

Miller W, Boguski MS, Raghavachari B, Zhang Z, Hardison RC. *Constructing aligned sequence blocks*. J Comput Biol 1994;1(1):51-64.

Miller W, Boguski MS. *A note about computing all local alignments*. Comput Appl Biosci 1994;10:455-456.

Mizushima H, Hayashi K. *Establishment of transcription factor database and human mutation database*. In: Proceedings of Genome Informatics Workshop 1994. (Miyano S, Akutsu T, Imai H, Gotoh O, Takagi T, eds.). Universal Academy Press, Tokyo, 1994.

Mortimer RK, Schild D, Contopoulou CR, Kans JA. *Genetic and physical maps of Saccharomyces cerevisiae*. Methods Enzymol 1991;194:827-863.

Muller M, Schnitzler P, Koonin EV, Darai G. Identification and properties of the largest subunit of the DNA-dependent RNA polymerase of fish lymphocystis disease virus: Dramatic difference in the domain organization in the family Iridoviridae. *J Gen Virol* 1995;76 (Pt. 5): 1099-1107.

Mushegian AR, Koonin EV. *Unexpected sequence similarity between nucleosidases and phosphoribosyltransferases of different specificity*. *Protein Sci* 1994;3:1081-1088.

Mushegian AR. *Genetic elements of plant viruses as tools for genetic engineering*. *Microbiol Rev*, Shepherd RJ, (in press).

Needleman SB, Wunsch CD. *A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins*. *Journal of Molecular Biology* 1970;48:443-453.

Neuwald A, Lawrence C, Liu J. *Gibbs motif sampling: Detection of bacterial outer membrane protein repeats*. *Protein Sci* 1995;4.

Noble JA, Innis MA, Koonin EV, Rudd KE, Banuett F, Herskowitz I. *The Escherichia coli hflA locus encodes a putative GTP-binding protein and two membrane proteins, one of which contains a protease-like domain*. *Proc Natl Acad Sci USA* 1993;90:10866-10870.

Nomenclature Committee of the International Union of Biochemistry (NC-IUB). *Nomenclature for Incompletely Specified Bases in Nucleic Acid Sequences*. *European Journal of Biochemistry* 1985;150:1-5.

Nussinov R, Pieczenik G, Griggs JR, Kleitman DJ. *Algorithms for Loop Matchings*. *SIAM Journal on Applied Mathematics* 1978;35:68-82.

Opipari AWJ, Boguski MS, Dixit VM. *The A20 cDNA induced by tumor necrosis factor alpha encodes a novel type of zinc finger protein*. *J Biol Chem* 1990;265(25):14705-14708.

Oppenheim AB, Rudd KE, Mendelson I, Teff D. *Integration host factor binds to a unique class of complex repetitive extragenic DNA sequences in Escherichia coli*. *Mol Microbiol* 1993;10(1):113-122.

Ossadnik SM, Buldyrev SV, Goldberger AL, Havlin S, Mantegna RN, Peng V-K, Simons M, Stanley HE. *Correlation approach to identify coding regions in DNA sequences*. *Biophysical Journal* 1994;67:64-70.

Ostell J. *Information issues [Book review]*. *Science* 1990;247:226.

Ostell J. *GenInfo ASN.1 syntax: sequences*. National Center for Biotechnology Information; Technical Report 1.

- Ostell J. *GenInfo backbone database overview*. National Center for Biotechnology Information; Technical Report 2.
- Ostell J. *GenInfo backbone schema summary*. National Center for Biotechnology Information; Technical Report 3.
- Osterburg G, Sommer R. *Computer Support of DNA Sequence Analysis*. Computer Programs in Biomedicine 1981;13:101-109.
- Pappu HR, Karasev AV, Anderson EJ, Pappu SS, Hilf ME, Febres VJ, Eckloff RM, McCaffery M, Boyko V, Gowda S, Dolja VV, Koonin EV, Gimpf DJ, Cline KC, Garnsey SM, Dawson WO, Lee RF, Niblett CL. *Nucleotide sequence and organization of eight 3' open reading frames of the citrus tristeza closterovirus genome*. Virology 1994;199:35-46.
- Peakman T, Crouzet J, Mayaux JF, Busby S, Mohan S, Harborne N, Wootton JC, Nicolson R, Cole J. *Nucleotide sequence, organization and structural analysis of the products of genes in the nirB-cysG region of the Escherichia coli K-12 chromosome*. Eur J Biochem 1990;191:315-323.
- Pearson WR, Lipman DJ. *Improved Tools for Biological Sequence Analysis*. Proc. Natl. Acad. Sci. USA 1988; 85: 2444-2448.
- Pearson WR. Rapid and Sensitive Sequence Comparison with FASTP and FASTA. In: *Methods in Enzymology*, (Doolittle RF, ed.), Academic Press, San Diego, California, USA, 1990;183:63-98.
- Pedersen AG, Baldi P, Brunak S, Chauvan Y. *Characterization of prokaryotic and eukaryotic promoters using hidden Markov models*. ISMB96 1996:182-191.
- Pesole G, Attimonelli M, Saccone C. *Linguistic analysis of nucleotide sequences: algorithms for pattern recognition and analysis of codon strategy*. Methods in Enzymology 1996;266:281-294.
- Popescu N, Landsman D, Bustin M. *Mapping the human gene coding for chromosomal protein HMG-17*. Hum Genet 1990;85:376-378.
- Prabhu VV, Young L, Prohofsky EW. *Observation of preferred direction melting in a lattice model of poly(dA-dC).poly(dG-dT)*. Biopolymers 1991;31(3):355-358.
- Prabhu V, Claverie J-M. *Correlations in intronless DNA*. Nature 1992;359:782.
- Prabhu VV. *Symmetry observations in long nucleotide sequences*. Nucleic Acids Research 1993;21:2797-2800.

Prestridge DS. *Predicting Pol II promoter sequence using transcription factor binding sites*. Journal of Molecular Biology 1995;249:923-932.

Prochnicka-Chalufour A, Casanova JL, Avrameas S, Claverie J-M, Kourilsky P. *Biased amino acid distributions in regions of the T cell receptors and MHC molecules potentially involved in their association*. Int Immunol 1991;3(9):853-864.

Rabiner LR. *A tutorial on Hidden Markov Models and selected applications in speech recognition*. Proc IEEE 1989;77(2):257-285.

Ramsingh A, Araki H, Bryant S, Hixson A. *Identification of candidate sequences that determine virulence in Cocksackievirus B4*. Virus Res 1992;23:281-292.

Recipon H, Perasso R, Adoutte A, Quetier F. *ATP synthase subunit c/III/9 gene sequences as a tool for interkingdom and metaphtyes molecular phylogenies*. J Mol Evol 1992;34(4):292-303.

Reuven NB, Koonin EV, Rudd KE, Deutscher MP. *Re gene for the longest known Escherichia coli protein and its relation to helicases*. J Bacteriol (in press).

Robberson BL, Cote GJ, Berget SM. *Exon definition may facilitate splice site selection in RNAs with multiple exons*. Mol Cell Biol 1990;10:84-94.

Rogozin IB, Milanese L, Kolchanov NA. *Gene structure prediction using information on homologous protein sequence*. Computer Applications in the Biosciences 1996;12(3):161-170.

Ross JI, Eady EA, Cove JH, Cunliffe WJ, Baumberg S, Wootton JC. *Inducible erythromycin resistance in staphylococci is encoded by a member of the ATP-binding transport super-gene family*. Mol Microbiol 1990;4:1207-1214.

Rudd KE, Miller W, Ostell J, Benson D. *Alignment of Escherichia coli K12 DNA sequences to a genomic restriction map*. Nucleic Acids Research 1990;18:313-321.

Rudd KE, Miller W, Werner C, Ostell J, Tolstoshev C, Satterfield SG. *Mapping sequenced E. coli genes by computer: software, strategies and examples*. Nucleic Acids Research 1991;19(3): 637-647.

Rudd KE. *Alignment of E. coli DNA sequences to a revised, integrated genomic restriction map*. In: Miller J, ed. A Short Course in Bacterial Genetics: a Laboratory Manual and Handbook for Escherichia coli and Related Bacteria. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1992;2.3-2.43.

Rudd KE, Schneider T. *Compilation of E. coli ribosome binding sites*. In: Miller J, ed. A Short Course in Bacterial Genetics: a Laboratory Manual and Handbook for Escherichia

coli and Related Bacteria. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1992;17:19-17.45.

Rudd KE, Bouffard G, Miller W. *Computer analysis of E. coli restriction maps*. In: Davies KE, Tilghman SM, eds. *Genome Analysis*; vol 4; *Strategies for Physical Mapping*. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1992;1-38.

Rudd KE. *Maps, genes, sequences, and computers: an Escherichia coli case study*. ASM News 1993;59(7):335-341.

Russel GJ, Walker PMB, Elton RA, Subak-Sharpe JG. *Doublet frequency analysis of fractionated vertebrate nuclear DNA*. J Mol Biol 1976;108:1-28.

Saitou N, Nei M. *The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees*. Molecular Biology and Evolution 1987;4:406-425.

Salamon P, Wootton JC, Konopka AK, Hansen LK. *On the robustness of maximum entropy relationships for complexity distributions of nucleotide sequences*. Comput Chem 1993;17:135-148.

Salzberg S. *Locating protein coding regions in human DNA using a decision tree algorithm*. Journal of Computational Biology 1995;2(3):473-486.

Salzberg S, Chen X, Henderson J, Fasman K. *Finding genes in DNA using decision trees and dynamic programming*. ISMB96 1996:201-210.

Sankoff D. *Efficient optimal decomposition of a sequence into disjoint regions, each matched to some template in an inventory*. Math Biosci 1992;111:279-293.

Schnitzler P, Janssen W, Sontag KC, Koonin EV, Darai G. *Identification of genes encoding Zn finger protein, non-histone chromosomal HMG protein homologue, and a putative GTP phosphohydrolase in the genome of Chilo iridescent virus*. Nucleic Acids Research 1994;22:158-166.

Schroeder JL, Blattner FR. *Formal Description of a DNA Oriented Computer Language*. Figure 1. Nucleic Acids Research 1982;10:69-84.

Schuler GD, Altschul SF, Lipman DJ. *A workbench for multiple alignment construction and analysis*. Proteins 1991;9(3):180-190.

Schuler GD, Epstein JA, Ohkawa H, Kans JA. *Entrez*. In: Doolittle R, ed. *Methods in Enzymology*. Academic Press, 1995; (in press).

Segev I, Fleshman JW, Burke RE. *Computer simulation of group Ia EPSPs using morphologically realistic models of cat alpha-motorneurons*. J Neurophysiol 1990;64:648-660.

- Senapathy P. *Introns and the origin of protein-coding genes*. Science 1995;268:1366-1367.
- Senkevich TG, Muravnik GL, Pozdnyakov VG, Chizhikov VE, Ryazankina OI, Shchelkunov SN, Koonin EV, Chernos VI. *Nucleotide sequence of Xho I O fragment of ectromelia virus DNA reveals significant differences from vaccinia virus*. Virus Res 1993;30:73-88.
- Senkevich TG, Koonin EV, Buller RML. *A poxvirus protein with a RING finger motif is of crucial importance for virulence*. Virology 1994;198(1994):118-128.
- Shah MB, Uberbacher EC, Einstein JR, Matis S, Xu Y, Buley D, Mural RJ. *Gene Recognition and Modeling in the GRAIL System*. Hilton Head DNA Sequence Conference, Hilton Head, SC, October 23-27, 1993.
- Shah MB, Matis S, Mural RJ, Uberbacher EC. *Detection of Human RNA Polymerase II Promoters Using Artificially Intelligent Methods*. First World Congress on Computational Medicine, Public Health, and Biotechnology, Austin, TX, April 24-28, 1993.
- Sharp PM, Li W-H. *The codon adaptation index: a measure of directional synonymous codon bias and its potential application*. Nucl Acids Res 1987;15:1281-1295.
- Sharp PM, Matassi G. *Codon usage and genome evolution*. Curr Opin Genet Dev 1994;4:851-860.
- Shin DG, Lee C, Zhang J, Rudd KE, Berg CM. *Redesigning, implementing and integrating Escherichia coli genome software tools with an object-oriented database system*. Comput Appl Biosci 1992;8(3):227-238.
- Shpaer EG. *Constraints on codon context in Escherichia coli genes; their possible role in modulating efficiency of translation*. J Mol Biol 1986;188:555-564.
- Sikorski RS, Boguski MS, Goebel M, Hieter P. *A repeating amino acid motif in CDC23 defines a family of proteins and a new relationship among genes required for mitosis and RNA synthesis*. Cell 1990;60:307-317.
- Sikorski RS, Michaud WA, Wootton JC, Boguski MS, Connelly C, Hieter P. *TPR proteins as essential components of the yeast cell cycle*. Cold Spring Harb Symp Quant Biol 1991;56:663-673.
- Smith DW, Jorgensen J, Greenberg JP, Keller J, Rogers J, Garner HR, Eyck LT. *Supercomputers, Parallel Processing, and Genome Projects*. In Biocomputing: Informatics and Genome Projects (Smith DW, Ed.), Academic Press, Inc., San Diego, CA, 1994: 51-86.

- Smith TF, Ralph WW, Goodman M, Czelusniak J. *Codon usage in the vertebrate hemoglobins and its implications*. Mol Biol Evol 1985;2:390-398.
- Smith TF, Waterman MS. *Comparison of Biosequences* Advan. Appl. Math. 1981;2: 482-489.
- Smith TF, Waterman MS. *Identification of Common Molecular Subsequences*. J. Mol. Biol. 1981;147:195-197.
- Sneath PHA, Sokal RR. Numerical Taxonomy. The Principles and Practice of Numerical Classification, W.H. Freeman and Company, San Francisco, California, USA, 1973.
- Snyder EE, Stormo GD. *Identification of protein coding regions in genomic DNA*. Journal of Molecular Biology 1995; 248:1-18.
- Solovyev VV, Salamov AA, Lawrence CM. *Prediction of human gene structure using linear discriminant functions and dynamic programming*. ISMB95 1995:367-375.
- Solovyev VV, Salamov AA, Lawrence CB. *Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames*. Nucl Acid Res 1994;22:5156-5163.
- Spouge JL. *Speeding up dynamic programming algorithms for finding optimal lattice paths*. SIAM J Appl Math 1989;49:1552-1556.
- Spouge JL. *Fast optimal alignment*. Comput Appl Biosci 1991;7(1):1-7.
- Spouge JL. Statistical analysis of sparse infection data and its implications for retroviral treatment trials in primates. Proc Natl Acad Sci USA 1992;89(16):7581-7585.
- Spouge JL. *Computation of the gamma, digamma, and trigamma functions*. SIAM J Numerical Anal 1994;31:931-944.
- Staden R. *Computer methods to locate signals in nucleic acid sequences*. Nucl Acids Res 1984;12:505-519.
- Staden R. *A New Computer Method for the Storage and Manipulation of DNA Gel Reading Data*. Nucleic Acids Research 1980; 8:3673-3694.
- States DJ, Boguski MS. *Similarity and Homology*. In: Gribskov M, Devereux J, eds. Sequence Analysis Primer. Salt Lake City: W.H. Freeman, 1991;89-157.
- States DJ, Botstein D. *Molecular sequence data accuracy and the analysis of protein coding regions*. Proc Natl Acad Sci USA 1991;88:5518-5522.

States DJ, Gish W, Altschul SF. *Improved sensitivity of nucleic acid database searches using application-specific scoring matrices*. Methods Companion to Methods Enzymol 1991;3:66-70.

States DJ, Harris NL, Hunter L. *Massive classification for automated annotation and domain identification in molecular sequence databases*. Genome Mapping and Sequencing. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1992:Abstract #191.

States DJ. *Molecular sequence accuracy: analysing imperfect data*. Trends Genet 1992;8(2):52-55.

States DJ, Gish W. *Combined use of sequence similarity and codon bias for coding region identification*. J Comput Biol 1994;1(1):39-50.

Stephens JC, Cavanaugh ML, Gradie MI, Mador ML, Kidd KK. *Mapping the human genome: current status*. Science 1990;250(4978):237-244.

Stephens JC, Mador ML, Cavanaugh ML, Gradie MI, Kidd KK. *The human genome map 1990*. Science 1990;250(4978):1-4.

Stepien PP, Margossian SP, Landsman D, Butow RA. *The yeast nuclear gene *suvs3* affecting mitochondrial post-transcriptional processes encodes a putative ATP-dependent RNA helicase*. Proc Natl Acad Sci USA 1992;89(15):6813-6817.

Sterner DA, Carlo, T, Berget SM. *Architectural limits on split genes*. Proc Natl Acad Sci USA 1996;93:15081-15085.

Stormo GD, Haussler D. *Optimally parsing a sequence into different classes based on multiple types of evidence*. In: Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology. AAI Press, Menlo Park, CA. 1994:47-55.

Sueoka N. *Directional mutation pressure, selective constraints, and genetic equilibria*. J Mol Evol 1992;34:95-114.

Sutherland GR, Richards RI. *Simple tandem DNA repeats and human genetic disease*. Proc Natl Acad Sci USA 1995;92:3636-3641.

Tang JL, Liu YN, Barber CE, Dow JM, Wootton JC, Daniels MJ. *Genetic and molecular analysis of a cluster of *rpf* genes involved in positive regulation of synthesis of extracellular enzymes and polysaccharide in *Xanthomonas campestris* pathovar *campestris**. Mol Gen Genet 1991;226(3):409-417.

Tatusov RL, Koonin EV, Lipman DJ. *A genomic perspective on protein families*. Science 1997;278:631-637.

- Tatusov RL, Koonin EV. *A simple tool to search for sequence motifs in BLAST outputs*. Comput Appl Biosci 1994;10:457-459.
- Tatusov RL, Altschul SF, Koonin EV. *Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks*. Proc Natl Acad Sci USA 1994;91:12091-12095.
- Tazi J, Bird A. *Alternative chromatin structure at CpG islands*. Cell 1990;60:909-920.
- Tiwari S, Ramachandran S, Bhattacharya A, Bhattacharya S, Ramaswamy R. *Prediction of probable genes by Fourier analysis of genomic sequences*. Submitted.
- Tolstoshev CM, Ashburner M, Kaufman TC, Mathews KA, Merriam JR, Ostell JM, Gelbart WM. *Overview of an integrated Drosophila database*. National Center for Biotechnology Information; Technical Report 7.
- Toone WM, Rudd KE, Friesen JD. *deaD, a new Escherichia coli gene encoding a presumed ATP-dependent RNA helicase, can suppress a mutation in rpsB, the gene encoding ribosomal protein S2*. J Bacteriol 1991;173:3291-3302.
- Trainor CD, Evans T, Felsenfeld G, Boguski MS. *Structure and evolution of a human erythroid transcription factor*. Nature 1990;343:92-96.
- Travers AA. **DNA-Protein Interactions**. Chapman & Hall, New York. 1993.
- Tugendreich S, Boguski MS, Seldin MS, Hieter P. *Linking yeast genetics to mammalian genomes: Identification and mapping of the human homolog of CDC27 via the expressed sequence tag (EST) data base*. Proc Natl Acad Sci USA 1993;90:10031-10035.
- Tugendreich S, Bassett DEJ, McKusick V, Boguski MS, Hieter P. *Genes conserved in yeast and humans*. Hum Mol Genet 1994;3:1509-1517.
- Turner DH. *Thermodynamics of base-pairing*. Curr Opin Struct Biol 1996;6:299-304.
- Uberbacher EC, Mural RJ. *Locating protein coding regions in human DNA sequences by a multiple sensor neural network approach*. Proc. of National Academy of Sciences USA 1991;88: 11261-11265.
- Uberbacher EC, Mural RJ, Guan X, Petrov S, Shah MB. *genQuest: A Sensitive Sequence Comparison Server for DNA and Proteins*. In: Genome Mapping and Sequencing, Cold Spring Harbor, NY, May 11-15, 1994.
- Uberbacher EC, Xu Y, Einstein JR, Mural RJ, Shah MB. *An Improved System for Exon Recognition and Gene Modeling in Human DNA Sequences*. The Second International Conference on Intelligent Systems for Molecular Biology, Stanford University, San Francisco, CA, August 1994.

- Uberbacher EC, Xu Y, Mural RJ, Shah MB. *Recognizing Exons in Genomic Sequence using GRAIL II*. Genetic Engineering, Principles and Methods, Plenum Press, Vol. 15, 1994.
- Uberbacher EC, Xu Y, Mural RJ. *Discovering and understanding genes in human DNA sequence using GRAIL*. Methods in Enzymology 1995; 266:259-281.
- Vasketich D. *Client/Server Strategies*. IDG Books, 1993.
- Viterbi, AJ. *Error bounds for convolutional codes and an asymptotically optimal decoding algorithm*. IEEE Trans Informat Theory. 1967;IT-13:260-269.
- Vranic-Sowers S, Fleshman JW. *Application of existing design software to problems in neuronal modeling*. IEEE Trans Biomed Eng 1994;41(3):291-294.
- Waldherr M, Ragnini A, Schweyen RJ, Boguski MS. *MRS6--yeast homologue of the choroideraemia gene [letter]*. Nature Genet 1993;3(3):193-194.
- Wang LP, Pichler EE, Ross J. *Oscillations and chaos in neural networks: an exactly solvable model*. Proc Natl Acad Sci USA 1990;87(23):9467-9471.
- Wang Y, Boguski MS, Riggs M, Rodgers L, Wigler M. *Sar1, a gene from Schizosaccharomyces pombe encodes a GAP-like protein that regulates ras1*. Cell Regul 1991;2:453-465.
- Wang D, Hubbard JM, Kabat EA. *Modeling study of the antibody combining sites to alpha(1->6)dextrans. Predictions of the conformational contribution of VL-CDR3 and J segments to groove-type combining sites*. J Biol Chem 1993;268:20584-20589.
- Waterman M. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman Hall, 1995.
- White CT, Singh RK, Reintjes PB, Lampe J, Erickson BW, Dettloff WD, Chi VL, Altschul SF. *BioSCAN: a VLSI-based system for biosequence analysis*. Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors. Boston: IEEE Computer Society Press, 1991:504-509.
- Wieringa B, Hofer E, Weissmann C. *A minimal intron length but no specific internal sequence is required for splicing the large rabbit B-globin intron*. Cell 1984;37:915-925.
- Wilbur WJ, Major F, Bryant S. *The statistics of unique native states for random peptides*. Biopolymers (in press).
- Wilbur WJ, Sirotkin K. *The automatic identification of stop words*. J Inf Sci 1992;18:45-55.

- Wilbur WJ. *An information measure of retrieval performance*. J Info Systems 1992; 17(4):283-298.
- Wilbur WJ. *Retrieval testing by the comparison of statistically independent retrieval methods*. J Am Soc Info Sci 1992;43(5):358-370.
- Wilbur WJ. *A retrieval system based on automatic relevance weighting of search terms*. In: Shaw D, ed. Proceedings of the 55th American Society of Information Science Annual Meeting. Pittsburgh, PA: Learned Information, Inc., 1992:216-220.
- Wilbur WJ. *Retrieval testing in hypergeometric document models*. J Am Soc Inf Sci 1993;44(6):340-351.
- Wilbur WJ, Coffee L. *The effectiveness of document neighboring in search enhancement*. Inf Process Manage 1994;30(2):253-266.
- Wilbur WJ, Liu JS. *Energy distribution of the compact states of a peptide chain*. Macromolecules 1994;27(9):2432-2438.
- Wilbur WJ. *Non-parametric significance tests of retrieval performance comparisons*. J Info Sci 1994;20(4):270-284.
- Wilbur WJ, Lipman DJ. *Rapid Similarity Searches of Nucleic Acid and Protein Data Banks*. Proceedings of the National Academy of Sciences USA 1983;80:726-730.
- Wilbur J. *A fully Bayesian cluster based model for document retrieval*. NCBI; Technical Report 6.
- Wolf H, Modrow S, Motz M, Jameson B, Hermann G, Fortsch B. *An Integrated Family of Amino Acid Sequence Analysis Programs*. Computer Applications in the Biosciences 1987; 4(1):187-191.
- Wong KK, Wong RM, Rudd KE, M. M. *High-resolution restriction map for a 240-kilobase region spanning 91 to 96 minutes on the Salmonella typhimurium LT2 chromosome*. J Bacteriol 1994;176:5729-5734.
- Woodsmall RM, Benson DA. *Entrez: Sequences and Entrez: References; NCBI's Genbank on CD-ROM*. Biocomm Data (in press).
- Woodsmall RM, Benson DA. *Information resources at the National Center for Biotechnology Information*. Bull Med Libr Assoc 1993;81(3):282-284.
- Wootton JC. *Simple Sequences of Protein and DNA*. In: Bishop MJ, Rawlings CR, eds. Nucleic Acid and Protein Sequence Analysis: A Practical Approach. 2d ed. Oxford: IRL Press, vol I, (in press).

Wootton JC, Federhen S. *Segmentation by sequence complexity: the significance of residue clusters in protein sequences (extended abstract)*. In: Salamon P, Konopka AK, eds. Open Problems in Computational Molecular Biology. Telluride, Colorado: Telluride Academy, 1992:77-81.

Wootton JC, Federhen S. *Statistics of local complexity in amino acid sequences and sequence databases*. Comput Chem 1993;17:149-163.

Wootton JC. *Non-alignability and non-compactness: models and reality of low-complexity protein segments (extended abstract)*. In: Salamon P, Konopka AK, eds. Open Problems in Computational Molecular Biology. Telluride, Colorado: Telluride Academy, 1993:65-68.

Wootton JC. *Sequences with 'unusual' amino acid compositions*. Curr Opin Struct Biol 1994;4:413-421.

Wootton JC. *Non-globular domains in protein sequences: automated segmentation using complexity measures*. Comput Chem 1994;18:269-285.

Wootton JC, Federhen S. *Taxonomy of simple amino acid sequences*. In: Lim HA, Cantor CR, eds. Bioinformatics and Genome Research. Singapore: World Scientific Publishing, 1995:159-172.

Wu, T. *A segment-based dynamic programming algorithm for predicting gene structure*. J Comp Biol 1996;3(3):375-394.

Xu Y., Uberbacher EC. *Gene prediction by pattern recognition and homology search*. ISMB96 1996: 241-251.

Xu Y, Einstein JR, Mural RJ, Shah M, Uberbacher EC. *An improved system for exon recognition and gene modeling in human DNA sequences*. In: Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology. AAI Press, Menlo Park, CA. 1994:376-384.

Yada T, Hirosawa M. *Gene recognition in Cyanobacterium genomic sequence data using the hidden Markov model*. ISMB96 1996:252-260.

Yang Y, Wilbur J. *Using corpus statistics to remove redundant words in text categorization*. J Am Soc Info Sci (in press).

Yarus M, Foley LS. *Sense codons are found in specific contexts*. J Mol Biol 1985;182:529-540.

Zhang J, Ostell J, Rudd KE. *Chromosome: A Graphic Interactive Browser for Biological Data Expressed in the NCBI Data Model*. In: Hunter L, ed. 27th Hawaii International

Conference on System Sciences. Maui, Hawaii: IEEE Computer Society Press, 1994:58-67.

Zhang MQ, Marr TG. *A weight array method for splicing signal analysis*. Comp Appl Biol Sci 1993;9(5):499-509.

Zhou S, Yanf Y, Scott MJ, Pannuti A, Fehr KC, Eisen A, Koonin EV, Fouts DL, Wrightsman R, Manning JE, Lucchesi JC. *Male-specific lethal 2, a dosage compensation gene of Drosophila, undergoes sex-specific regulation and encodes a protein with a RING finger and a metallothionein-like cysteine cluster*. EMBO J 1995;14:2884-2895.

Zuker M. *Computer Prediction of RNA Structure*. In: Methods in Enzymology , (J.E. Dahlberg and J.N. Abelson, eds.), Academic Press, San Diego, California, USA, 1989; 180:262-288.

Zuker M. *On Finding All Suboptimal Foldings of an RNA Molecule*. Science 1989; 244:48-52.

Zuker M, Stiegler P. *Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information*. Nucleic Acids Research 1981;9:133-148.