Functional Requirements of Biosequence Annotation Systems

Melissa Lemos
e-mail: melissa@inf.puc-rio.br

Luiz Fernando Bessa Seibel
e-mail: seibel@inf.puc-rio.br

Marco Antônio Casanova
e-mail: casanova@inf.puc-rio.br

PUC-RioInf.MCC03/04 Janeiro, 2004

Abstract: One of the most important tasks of genome projects is the interpretation of experimental data in order to derive biological knowledge from the data. To achieve this goal, researchers typically search external data sources, execute analysis programs on the biosequences, analyze existing annotations and add new annotations to register their interpretation of the data.
This paper first summarizes selected tools and techniques used in Bioinformatics. Then, it lists functional requirements for biosequence annotation systems. It proceeds to describe the functionality and the data model of BioNotes, a tool that meets these requirements. Next, it outlines how the tool will support workflow-like composition of analysis programs. Finally, it compares BioNotes with other annotation systems.
BioNotes is under development at the Catholic University of Rio de Janeiro and in use at Oswaldo Cruz Institute and at RioGene.

Keywords: Annotation systems; Bioinformatics; Database; BioNotes.

Resumo: Uma das principais tarefas dos projetos genoma é a interpretação de dados experimentais para se obter conhecimento biológico a partir do dado. Para alcançar este objetivo, os pesquisadores geralmente fazem buscas em fontes de dados externas, executam programas de análise nas biossequências, analisam as anotações existentes e adicionam novas anotações que registram sua interpretação sobre os dados.
Esta monografia apresenta uma visão geral sobre os programas de análise e as fontes de dados externas existentes em bioinformática e lista os requisitos funcionais de sistemas de anotação de biossequências. O sistema de anotação BioNotes é apresentado dando ênfase a descrição de como é tratada a composição de programas de análise. Finalmente, é apresentado uma comparação entre os principais sistemas de anotação existentes.

Palavras-chave: Sistemas de anotação; Bioinformática; Banco de Dados; BioNotes.

1 Introduction

This introduction first provides a broad perspective of Bioinformatics. It touches on how genome projects are conducted, the computational tools and techniques they use, the characteristics of Molecular Biology databases and the major features of annotation systems. The introduction concludes with the objectives and overall organization of the paper.

Genome Projects

The ultimate goal of the Human Genome Project is to generate a high-quality reference DNA sequence for the human genome – some 3 billion base pairs – and to identify all human genes. Other important goals include sequencing the genomes of model organisms to interpret human DNA, enhancing computational resources to support future research and commercial applications, exploring gene function through mouse-human comparisons, studying human variation, and training future scientists in genomics [57].

Each cell of a living organism contains chromosomes composed of a sequence of DNA base pairs. This sequence, the genome, represents a set of instructions that controls the replication and function of each organism.

Although genomes vary in size from millions of nucleotides in bacteria to billions of nucleotides in humans and most animals and plants, the chemical reactions researchers use to decode the DNA base pairs are accurate for only about 600 to 700 nucleotides at a time.

The process of sequencing begins by physically breaking the DNA into millions of random fragments, which are then "read" by a DNA sequencing machine. Computer analysis is then used to build chromatograms (consisting of four curves of different colors, each curve representing the signal for one of the four bases), and to convert the chromatogram to an inferred base sequence (or *read*). Phred [35,36] is an example of program which does this task.

Next, a computer program called an *assembler* (for example, CAP3 [34] and Phrap [37]) pieces together the many overlapping reads and reconstructs the original sequence. This general technique is called *shotgun sequencing*. This process is not simple because the data contains errors – some from limitations in sequencing technology and others from human mistakes during laboratory work. Even in the absence of errors, DNA sequences have features that complicate the assembly process, like repetitive sections called *repeats*.

In practice, imperfect coverage, repeats and sequencing errors cause the assembler to produce not one but hundreds or even thousands of *contigs*. The task of closing the gaps between contigs and obtaining a complete molecule is called *finishing* [58].

Because genome projects generate raw data without giving them biological meaning, there is another process, called *annotation process*, which has the task of converting experimental data (raw DNA data) into biologically relevant information (annotated sequences) [59].

The annotation is a meta-information or a description of high level features of the biological sequence, or simply, biosequence. Useful information includes whether a stretch of DNA contains an amino acid coding sequence, the *transposons*, or a regulatory sequence, and, if an amino acid is coded, what its putative function is, and so on [59].

Researchers use a variety of computer tools or programs, combined with human interpretation, to carry out much of this annotation. However, given the rate at which researchers now generate DNA sequence information, automatically annotating the raw data presents a computational challenge, and careful human analysis is becoming increasingly difficult.

Computational Tools and Techniques

When researchers isolate new molecular sequence data in the laboratory, they want to know everything about that sequence. A first step is to see if other researchers have already studied any molecular sequences similar to this sequence. Probably the most widely used computational tool in biology, BLAST [33] – Basic Local Alignment Tool – searches databases like Genbank [1] for all sequences similar to a target sequence. Indeed, when researchers isolate a new molecular sequence, the first thing they usually do is run a BLAST search against existing databases [59].

Alignments provide a powerful way to compare related sequences. An alignment can be either global or local, depending on the purpose of the comparison. Global alignment forces complete alignment of the input sequences, whereas local alignment detects only their most similar segments.

The first algorithm designed to detect the optimal global alignment was the Needleman-Wunsch algorithm. Subsequently, a slight variant was proposed, termed the Smith-Waterman algorithm [53], which can find the optimal local alignment of two sequences. Both these algorithms require time proportional to the product of the lengths of the sequences being compared.

Because similarities between DNA and protein sequences often span only segments of the sequences involved, the most popular similarity search programs are based on the Smith-Waterman local alignment algorithm. However, without special-purpose hardware or massively parallel machines, the time required by Smith-Waterman proved to be too slow for most users. The FASTA [54] and BLAST programs, therefore, use heuristics to concentrate their efforts on the sequence regions most likely to be related [60].

Global and local pairwise sequence comparison and alignment can be generalized to multiple sequences. ClustalW [51] is an example of a multiple sequence alignment algorithm.

In addition to sequence comparison, tools are required for gene prediction, gene classification, comparative genomics, structure prediction, phylogenetic analysis, pattern discovery, pattern recognition and others, as briefly described below.

GLIMMER [2] and ORF Finder [77] are example of gene prediction programs. PHYLIP [61] (the PHYLogeny Inference Package) and PAUP [62] (Phylogenetic Analysis Using Parsimony) are the most popular phylogeny packages. TRIPOS [63] is a structured prediction program.

To allow a direct comparison of the genomic sequences of sufficiently similar organisms, there is an urgent need for software tools that can align more than two genomic sequences. Alfresco [65] and MGA [66] (Multiple Genome Aligner) are examples of programs designed for multi-genome comparison.

Nucleotide and amino acid sequences contain patterns that have been preserved through evolution because they are important to the structure or function of the molecule. In proteins, these conserved sequences may be involved in the binding of the protein to its substrate or to another protein, may comprise the active site of an enzyme or may determine the three dimensional structure of the protein. Nucleotide sequences outside of coding regions in general tend to be less conserved among organisms, except where they are involved in the regulation of gene expression. Discovery of patterns in protein and nucleotide sequences can lead to determination of function and to elucidation of evolutionary relationships among sequences [67]. Examples of pattern discovery algorithms are Teiresias [68], Pratt [69] and Blocks Maker [70].

In pattern discovery, the algorithm is supposed to discover pattern unknown in advance. However, in Biology, many consensus sequences are known. Therefore, it is important to have tools that find occurrences of known patterns in new sequences [67]. This problem is called pattern matching. RBSFinder [39] is an example of pattern matching program which finds ribosome sites.

Molecular Biology Databases

Molecular biology databases have been proliferating rapidly. Some databases concentrate on specific molecules or specific functions and provide highly detailed information, while others try to cover a broad range of biology with less detailed information. In some cases, the biological information is generated by computer analysis of other databases; in other cases, it is obtained from the literature by manual means. Genbank is an example of nucleotide sequence database and SWISS-PROT [30] is a curated protein sequence database. PROSITE [31] is derived from SWISS-PROT and contains conserved sequence patterns associated with specific functions. There are other pattern libraries, notably BLOCKS [71] and PRINTS [72], as well as

libraries containing longer sequence patterns of protein domain structures, such as Pfam [73] and ProDom [74]. One of the main differences is how the sequence information is represented, namely in regular expression of text patterns, multiple alignmnents, profiles or hidden Markov models [75].

As we can see, Biology is a diverse field that collects information from many distributed sources. Given these heterogeneous information sources, collecting and integrating them into a coherent information set presents a huge problem. An example of a challenge is the seemingly trivial problem of how to define a gene.

Indeed, a researcher now has access to a rich set of data sources, as well as a variety of data analysis programs. The public data sources are heterogeneous, feature large data volumes, are in constant growth, but do not usually have a detailed documentation of the database schema. Also, the analysis programs do not have good documentation that describes the execution parameters and the input and output data formats, which makes it difficult to integrate them into more complex workflow processes.

Annotation Systems

As a consequence, in the annotation process, the challenge of Bioinformatics is to create effective tools to help researchers mine large sets of biosequences, that is, sets of DNA or protein sequences. This involves many facets. The annotation systems must help a researcher: access annotations stored in public data sources; execute analysis programs to obtain automatic annotations, analyze current annotations, with the help of an appropriate interface, and manually generate new annotations.

From the scenario just described, we may then classify annotations as *manual*, directly created by the researcher, *automatic*, generated by analysis programs, or *imported* from public data sources.

The characteristics of the annotations also vary according to the goal of the genome project. Indeed, annotations generated in the context of a project that targets the complete DNA of an organism have different requirements from those created in the context of a project whose goal is to obtain ESTs (expressed sequence tags) [57], which is common for large genomes.

Objectives and Organization of the Paper

The objectives of this paper are to elicit the functional requirements of biosequence annotation systems and to describe BioNotes, a tool that meets these requirements. BioNotes is under development at the Catholic University of Rio de Janeiro and is currently used by the Department of Biochemistry and Molecular Biology [4], Oswaldo Cruz Institute – FIOCRUZ, to annotate the genome of *Trypanosoma cruzi*, and by RioGene [5] (a Virtual Institute formed by universities and research institutes from the Rio de Janeiro state), to annotate the genome of the bacteria *Gluconacetobacter diazotrophicus.*

The paper is organized as follows. Section 2 presents the major functional requirements of annotation systems, based on an analysis of the major annotation systems available. Section 3 describes BioNotes. Section 4 presents a discussion about how BioNotes will support will support workflow-like composition of analysis programs. Section 5 presents a comparison with related work. Finally, section 6 contains the conclusions and directions for future work.

2 Functional Requirements of Annotation Systems

The goal of annotation systems is to help researchers create, retrieve and analyze annotations that tag biosequences.

The functional requirements of annotation systems listed in this section result from a careful study of the following systems: Artemis[6], DAS[7], CeleraBrowser [8], EDITtoTrEMBL[9], GASP[10], GenDB [11], GeneMine [12], GeneQuiz [13], Apollo [14], Gbrowser [15], Imagene [16], MAGPIE [17], Manatee [18], Pedant [19], VisualGenome [20], PseudoCAP [21], Community Annotation Project [22], Alternative Splicing Annotation Project [23], Genestream [24], Cancer Annotation Project [25], Ensembl Genome Annotation Project[26] and NCBI's Genome Annotation Project [27].

The major functional requirements for annotation systems, along with brief comments, are listed below. The requirements are organized hierarchically to facilitate understanding their purpose.

R1. To model persistent annotations:

R1.1. To model *external annotations*, persisted in external data sources, not under control of the annotation system. To meet this requirement, it becomes necessary to understand the external data sources, which are heterogeneous, store large volumes of data and are in constant growth, and to understand their schemas, which are often not well documented.

R1.2. To model *internal annotations*, persisted in a data warehouse under control of the annotation system.

R1.2.1. To model annotation metadata attributes, such as author, source and creation date, consistently with Dublin Core Metadata Element Set [76].

R1.2.2. To model *imported annotations*, copied from external data sources to the data warehouse. To meet this requirement, it again becomes necessary to understand the external data sources.

R1.2.3. To model *automatic annotations*, generated by executing applications and persisted in the data warehouse. To meet this requirement, it becomes necessary to understand the applications, which sometimes present problems during their execution, and to understand their execution parameters, which are often not well documented.

R1.2.4. To model *manual annotations*, created by researchers and persisted in the data warehouse. To meet this requirement, it becomes necessary to offer a well designed, well documented user interface.

1.2.4.1. To offer a controlled vocabulary to create manual annotations, such as gene and organisms ontologies. This requirement guarantees better control over manual annotations, contributing to their quality.

R1.3. To offer extensibility mechanisms to model new external or internal annotation types. This requirement facilitates covering a wide variety of organisms and project configurations (complete genome projects, EST identification projects, etc.).

R2. To offer tools to access annotations:

R2.1. To offer tools to access external annotations.

R2.2. To offer tools to query internal annotations and to display the results in an appropriated format. The tools must be accessible to a distributed user community through a well designed, well documented user interface.

R2.2.1. To offer tools to graphically display internal annotations, genome elements and their relationships (for example, the user must be able to visualize a *contig*, with the associated *reads,* and to identify the ORFs found in the *contig*).

R2.2.2. To offer tools to display internal annotations in tabular format. This requirement facilitates to visually inspect and compare annotations of the same genome element, but coming

from different sources.

R3. To offer tools for annotation versioning.

R3.1. To periodically re-import annotations from the external data sources to the data warehouse. This requirement is well justified since external sources are constantly updated with potentially useful new annotations.

R3.2. To periodically re-generate annotations persisted in the data warehouse and created by executing applications, whenever new data becomes available.

R3.3. To automatically transfer manual annotations persisted in the data warehouse from older to newer versions of the data (see remarks at the end of the list).

R4. To offer tools to control the execution of applications:

R4.1. To offer remote execution of applications residing in external sites.

R4.2. To offer local, offline execution of applications, persisting the results in the data warehouse. This requirement facilitates fast access to the application results, but obviously requires that the input data be available in advance. Also, persisting the results of offline execution is beneficial when a community of users plans to access, visualize or analyze the same (derived) data repeatedly.

R4.3. To offer local, online execution of applications. This requirement permits a single user to run an application for his own use, perhaps with newly obtained data. Online execution is the right option when the user does not want, or see no reason for sharing the results outside his transaction.

R4.4. To offer extensibility mechanisms to accommodate new applications. This requirement facilitates covering a wide variety of organisms and project configurations (complete genome projects, EST identification projects, etc.).

R4.5. To offer flexible application composition, using workflow concepts.

R5. To control access to the system and to the data stored in the data warehouse. This requirement is necessary to define data sharing policies among communities of users, specially in the context of genome projects whose data is not of public domain. It reflects the concept of a Grid Computing user community.

R6. To offer tools to generate reports that help manage the overall project (for example, for on-going genome projects, it is interesting to obtain information about the last fragment assembly executed, showing the identification, size and quality of the bases and the number of *reads* in each *contig*).

R7. To support discussion fora, where users may exchange information about: (1) the annotation system itself; (2) the development of their projects; and (3) research topics, in general (see remarks at the end of the list).

Requirement R3.3 is specially useful in the context of on-going genome sequencing projects. Indeed, in this case, while new *reads* are sequenced, fragment assembly programs, such as Phrap and CAP3, will probably generate totally new *contigs*, or modify already existing *configs*. Thus, programs that process *contigs*, such as Glimmer and tRNAScan, may also generate new annotations in the new or modified contigs. Moreover, the annotations researchers manually added to the original *contigs* must automatically migrate to the new or

modified *contigs*. Therefore, Requirement R3.3 will make it feasible to manage annotations in the context of on-going genome sequencing projects, thereby permitting early release of the data. That is, it will permit researchers to proceed to the interpretation stage, without having to wait for the complete sequencing of the genome.

Requirement R7 also deserves explanation. Exchanging information about the annotation system is indeed important since the number of applications and external data sources that can be aggregated to the system is very large. Therefore, researchers should use the discussion forum to reach an agreement about what applications should be used at each stage of the project, how to set their parameters and what data sources should be accessed. Now, exchanging information about the development of the projects becomes relevant when annotations are added in parallel with sequencing and fragment assembly. Indeed, such annotations may be used to help the sequencing and fragment assembly processes themselves.

The systems investigated do not meet all the requirements listed. For example, only one of the systems, Imagene [16], permits a workflow-like composition of analysis programs, but it does not offer distributed input of manual annotations. Section 5 contains an extensive comparison between various annotation systems, including BioNotes.

3 The BioNotes System

The BioNotes systems, briefly described in this section, was designed to meet the requirements listed in Section 2.

The architecture of the system follows the MVC (model-view-controller) design pattern [55,56], which separates the application into three modules: the business logic part, which implements data retrieval and manipulation; the user interface part; and the controller part, which routes requests to the proper objects.

Its implementation uses the infrastructure of the Bio-AXS system [28], a biology data warehouse integrating data from several public sources. BioNotes is written in Java 1.4.0, for portability, and offers a Web interface implemented with JSP technology. The interface is certified for Internet Explorer 5.0 or later. The tool runs on top of a relational database management system, which directly stores XML documents and supports Xpath queries.

3.1 Functional Description of BioNotes

BioNotes implements the concept of a *user community* to control access to the data. A user community is just a set of users, possibly from different institutions. The system also offers different user profiles to further control which users can execute what commands.

BioNotes supports all three types of annotations - imported, automatic and manual – and several types of biosequences - *contigs*, *reads* and *singletons*, among others.

A user may currently search annotations imported from the following external public sources: GenBank [1], PIR [29], SWISS-PROT [30], PROSITE [31] and Interpro [32].

The user may also run and store the annotations automatically generated by the following analysis programs: Phred [35,36], Phrap [37], GLIMMER [2], tRNAScan [3], RBSFinder [39], transTerm [40], BLAST [33], InterproScan [32] and CAP3 [34].

BioNotes transforms the annotations imported from external data sources, or automatically generated by analysis programs, to a common format – expressed as a XML schema – before storing them in the data warehouse (see Section 3.3).

Moreover, the user may manually add, delete and update annotations, which then become available to his community. Therefore, the concept of a user community facilitates sharing annotations in a controlled way. Updating a manual annotation actually creates a new version of the annotation. Hence, the user may browse through the various versions of an annotation and track down who created them. Only the user who created (a version of) an annotation may delete it.

BioNotes offers a variety of tools to access the annotations stored in the data warehouse. For example, given a biosequence identifier, the user may retrieve all annotations pertaining to the biosequence, such as the

keywords and feature table assigned to the biosequence by Genbank. Likewise, the user may retrieve all annotations automatically assigned to the biosequence by the execution of an analysis program, such as the homologous biosequences obtained by running BLAST. In addition, BioNotes offers graphical schemes for data visualization. For example, the system graphically indicates, for a contig *C*, the reads that compose *C* and the ORFs, tRNAs and ribossome sites present in *C*. These graphical elements are links to their respective annotations.

The system also has a color coding scheme to indicate the quality of a base (that is, A, T, C or G in DNA sequences) that helps the user analyze the quality of the read.

BioNotes also permits the user to access biosequence annotations stored in other sites, such as NCBI [41]. For example, the result of executing BLAST to compare a sequence *S* with the NR database [42] (obtained from NCBI) will contain links to the NCBI site that point to the annotations of each biosequence that is similar (according to BLAST) to *S*.

The system supports the concept of a *private data source*, that is, a local set of biosequences and annotations that is private to a user community. The current examples are: TCRUZI, a private data source that stores annotations and biosequences pertaining to the *Trypanosoma cruzi* organism; and GLUCONA, that stores annotations and biosequences pertaining to the *Gluconacetobacter diazotrophicus* organism.

These two private data sources were created using very different strategies. TCRUZI exemplifies a private data source created by importing data from a public data source, Genbank in this case. Indeed, data pertaining to the *Trypanosoma cruzi* organism were first extracted from Genbank, which is not difficult since Genbank lets one search the data by organism name. Then, the data were remapped to an internal format, defined by an XML schema stored in BioNotes (see also Section 3.3), and stored in the data warehouse. By contrast, GLUCONA was directly created using BioNotes tools. The chromatograms of the *Gluconacetobacter diazotrophicus* organism that the sequencing lab made available were first submitted to the Phred program, which generated *reads* and annotations. These data were then remapped to an internal format, again defined by an XML schema stored in BioNotes, and stored in the data warehouse.

The user may analyze biosequences by executing programs at sites external to BioNotes. For example, the NCBI site permits executing BLAST to compare a sequence with several databases. BioNotes allows the user to compare a biosequence with several databases stored at NCBI by sending the sequence to the NCBI server and invoking the BLAST service at the site.

Molecular Biology data sources are prone to errors and inconsistencies. To facilitate data analysis, BioNotes tries to store curated (consistent) data from the various sources, such as SWISS-PROT, and to execute programs, such as BLAST, using non-redundant data sources, such as NR, obtained from NCBI.

BioNotes stores, together with each annotation, its source. However, this tracing is inefficient when the annotation originates from a database that has not been curated. Indeed, such data source guarantees neither data quality nor indicates how the annotation was obtained.

We conclude this section with a challenging issue. While a genome project is in progress, sequencing of the genome will generate new *reads*. As a consequence, the data sources will constantly be updated and new runs of the analysis programs will further generate new derived data. If the data source is external, to remain up-to-date, BioNotes must refresh its data warehouse from time to time by re-accessing the data sources to retrieve new data. Also, the system must locally re-run the analysis programs to generate new versions of the derived data. Therefore, the system must offer tools to help users compare different versions of the data, as new reads become available, and to transfer manual annotations from older to newer versions. This issue is further discussed in Section 6.

3.2 The Data Model of BioNotes

BioNotes features a semi-structured data model, implemented on top of relational tables, with columns storing semi-structured data as XML documents. The system keeps the description of the XML documents as XML schemas, again stored in a special table.

The system currently features:

- pre-defined functions to query XML data;
- special indexes over the XML data to improve performance;
- pre-defined functions to check the syntax of XML documents against XML schemas.

The entity class *Annotation* (see Figure 1) is specialized into the sub-classes *Automatic, Imported* and *Manual*. All annotations are implemented as XML documents stored in table columns (of type XML). The entity class *Schema* contains the XML schemas that define the syntax of the XML documents that represent annotations.

For each data source D_r, the entity class *Data Source* D_r (see Figure 2) represents the annotations imported from D_r. Likewise, for each analysis program P_q, the entity class *Analysis Program* P_q (see Figure 3) represents the annotations automatically generated by P_q.

Finally, the entity classes *User*, *Community* and *Institution* (see Figure 4) model the concept of a user community.

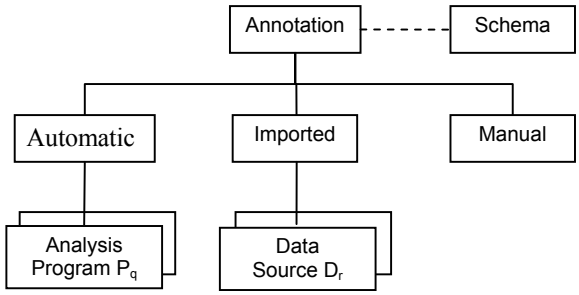

Figure 1. The *Annotation* entity class.

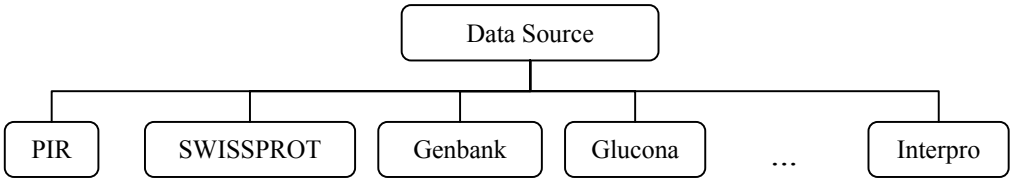

Figure 2. Data Sources.

Certain external data sources, such as PIR, Interpro, SWISS-PROT and Genbank, already distribute data as XML documents, whose structure is also available (either as XML schemas or DTDs). BioNotes then stores data exported from theses sources in their external format. However, other data sources, such as Prosite and Blocks, distribute their data in a proprietary format. In these cases, BioNotes has specific XML schemas to define the local format of the data and offers parsers to transform their external format to the BioNotes format.

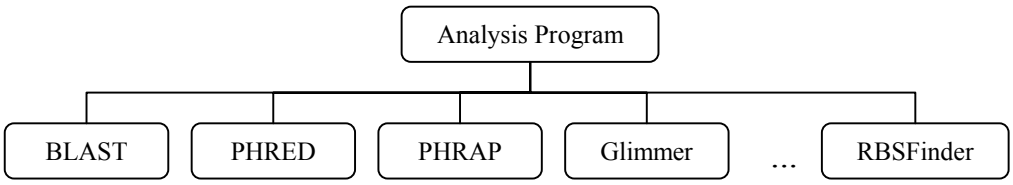

Figure 3. Biosequence analysis programs.

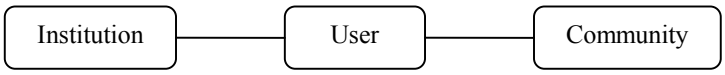

Figure 4. Users, Communities and Institutions.

Likewise, certain analysis programs generate data in XML format according to a well-defined (and published) XML schema, while others generate data in a proprietary format. InterproScan and BLAST are examples of former variety, while Phred, Phrap, GLIMMER, tRNAScan and RBSFinder of the latter variety. BioNotes also offers XML schemas and parsers to transform output data from theses programs into XML documents, before storing them into the data warehouse.

Finally, manual annotations are also stored as XML documents that follow an XML schema. When creating an annotation, the user is offered a Web form, where certain fields have a controlled vocabulary, while others are of free format. The form is first parsed into an XML document, which is then stored.

4 Workflow Support

4.1 Workflow in the context of biosequence annotation systems

In the context of this paper, a *workflow* [43, 44, 45] defines a composition of the programs described in Section 3.3. We assume that workflows are written in a suitable *workflow language*, such as XPDL [46], which we leave unspecified for the sake of brevity. In what follows, we focus on the peculiarities of the workflow language the BioNotes system implements. We will also use the term 'service' to refer to a program implemented as a Web service by some other system.

First, a *concrete* program is just a program that is actually implemented by the Bionotes system. An *abstract* program is a generic name for a set of concrete programs or, recursively, a set of abstract programs so that abstract and concrete programs form a strict hierarchy.

A workflow is *abstract* if it refers to at least an abstract program; otherwise it is *concrete*. By specifying an abstract program in a workflow, the user leaves it for the system to automatically decide what is the best concrete program to use, perhaps based on quality parameters he specifies. He may also direct the system to help him select the concrete program that best matches his needs. These issues are further discussed in Section 4.2.

The roots of the program hierarchy in BioNotes are:

Read Identification (from Chromatograms)
DNA Fragment Assembly
Sequence Alignment
Genome Comparison
Secondary and Tertiary Structure Prediction
Phylogenetic Analysis

Pattern Discovery
Pattern Recognition
Gene Prediction

For example, the program hierarchy rooted at Sequence Alignment abstract program is (concrete programs are listed in italics):

Sequence Alignment
- Multiple Alignment
  - *CLUSTALW* [51]
  - *MultiAlin* [52]
- Pairwise Alignment
  - Global Alignment
    - Smith Waterman [53]
      - *ssearch*
  - Local Alignment
    - FAST [54]
      - *FASTA*
      - *FASTY3*
      - *FASTX3*
      - *TFASTA3*
      - *TFASTX3*
      - *TFASTY3*
    - BLAST
      - *BLASTP*
      - *BLASTN*
      - *BLASTX*
      - *TBLASTX*
      - *TBLASTN*

Now, a workflow may be executed *offline* or *online*. When the user submits a workflow W for offline execution, the system executes W, without user request or intervention, and persists the resulting data in the data warehouse. Naturally, this option is beneficial when a community of users plans to access, visualize or analyze the same (derived) data repeatedly.

By contrast, when the user submits W for online execution, the system immediately executes W, offers the resulting data for user analysis or visualization, and does not persist the data outside the scope of the user transaction. This type of workflow execution should be selected when the user does not want, or see no reason for sharing the results outside his transaction.

Figure 5 shows an example of a workflow, used in the *Glucona* Project as an offline workflow.

The first step invokes Phred and Phrap to generate the reads and contigs from the chromatograms of the Glucona bacteria. The chromatograms are obtained from the Riogene laboratories.

The following step executes a parser to transform the output of Phred and Phrap (reads and contigs) into XML documents and stores them in the data warehouse.

The main objective of the next step is to discover important patterns in the genome of the *Gluconacetobacter diazotrophicus* bacteria, such as Open Read Frames (ORFs) or putative genes, transfer RNAs (tRNAs), ribosome sites (RBSs) and terminators.

Gene discovery can be carried out using different programs, such as Glimmer and ORF Finder [77]. BioNotes currently executes Glimmer. tRNA discovery can, in turn, be done using tRNAScan-SE. The results of both of these programs are converted into XML documents and stored in the data warehouse.

The results produced by Glimmer, i.e. ORF sequences, are then used to discover terminators and ribosome sites. Terminator prediction is made by executing the Transterm [40], and ribosome site prediction, by

executing the RBSFinder [39]. Once more, the results of these programs are converted into XML documents and stored in the data warehouse.

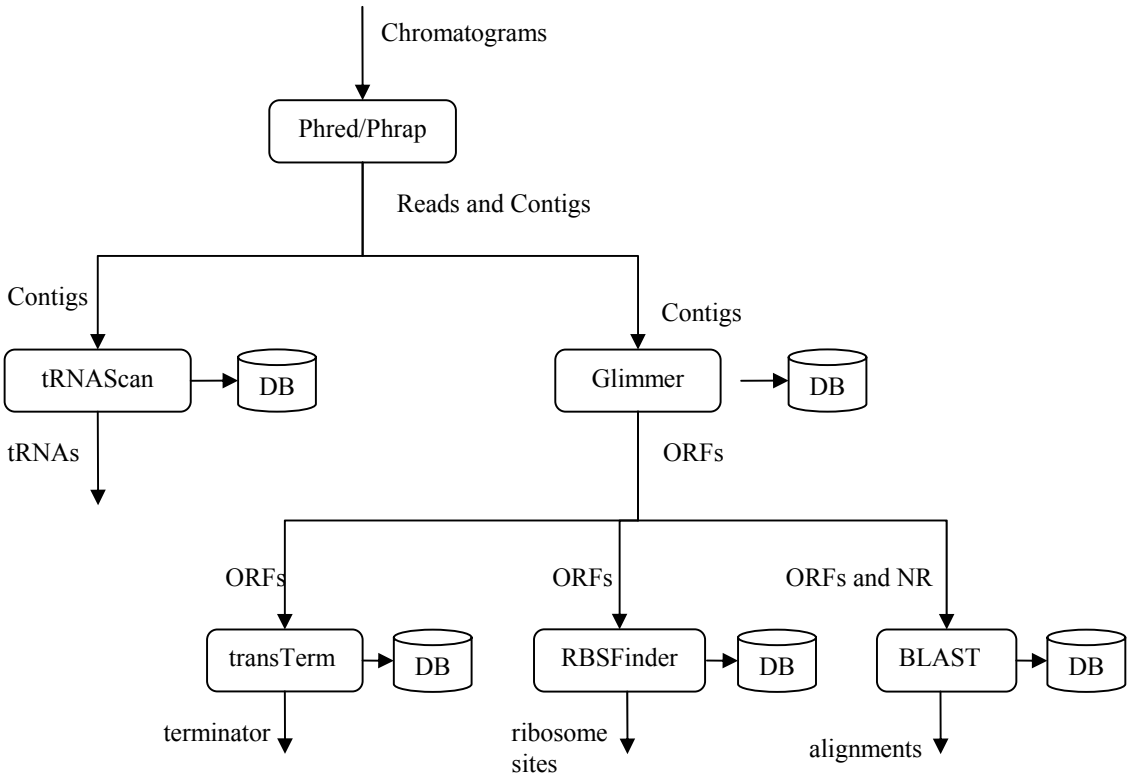

Figure 5 – Example of offline workflow execution for the Glucona Project.

It is important to note that some programs do not generate all the results that the research expects. For example, Transterm does not generate the terminator sequences themselves. Likewise, RBSFinder does not generate the RBS sequences. In these cases, it is necessary to execute a program called Extract [2], which uses start and end positions, passed as input, to extract sequences from the genome or contig.

Similarly, Glimmer generates ORF nucleotide sequences, and not amino acid sequences, so it is necessary to execute another program to translate nucleotide to amino acid sequences. Out of the many choices, BioNotes uses EMBOSS/Transeq [38].

Another important task is the comparison between ORFs and the public data source NR [42], which is an amino acid sequence database that can be downloaded from the NCBI site. The comparison is made using BLASTP. First, the ORF nucleotide sequences, generated by Glimmer, are converted into amino acid sequences using Transeq. Then, BLASTP is run. Finally, the results are stored in the data warehouse.

4.2 Helping user define workflows

To help users define workflows, in addition to a friendly interface, the annotation system should guide the user in his/her choice of program composition or specialization (from abstract to concrete programs). A common implementation strategy is to equip the system with a knowledge base that captures the major data and program characteristics, including quality parameters. An alterative approach is to define an ontology for the same purpose and design the system around it [47], thereby creating an *ontology-driven biosequence*

annotation system. The two approaches are essentially the same and will not be further distinguished.

The system may use the knowledge base in various ways. First, it may use the knowledge base to verify the coherence of the program composition the user is defining. That is, the system may verify if the output of one program is of the type expected by the next program. If this simple type checking fails, the system may suggest using a type conversion utility to make program composition viable. The system may also use the knowledge base to automatically select the best program or to help the user manually select the concrete program that best matches his needs, with the help of quality parameters.

For example, a user may define a local alignment of a sequence against a given database. This abstract program may be implemented by variants of BLAST or FAST, depending on the quality parameters the user determines. If the user privileges efficiency, the system should use BLAST. If he emphasizes precision, the system should choose FAST.

Suppose the system chooses BLAST. There are several parameters that the system must set, again according to the user directives. Depending on these parameters, the frequency of "false positive" results (false similarities found) and of "false negatives" (similarities not found) vary. These parameters must reflect the value of a fidelity parameter. Finally, since there are several implementations of these two services on the Web, the final choice may take into account an availability parameter.

In general, *program quality parameters* include:

performance	measures the amount of computational resources consumed by the program.
popularity	measures the size of the user community that knows and uses the program.
cost	measures the (financial) cost of running the program.
fidelity	measures the percentage of "false positives" and "false negatives" (applicable only to certain classes of programs).
default	indicates if the program is the default option for the program hierarchy it belongs to.
availability	measures the percentage of time a service is available (in the context of a program implemented as a Web service).

The value of these quality parameters may evolve over time. For example, the system may automatically update the knowledge base by learning new facts. Therefore, if the system detects that a concrete program is chosen with a high frequency, its popularity is updated.

Data quality parameters are somewhat more difficult to define and are tightly related to the hierarchy of abstract programs. For example, focusing on Fragment Assembly, the type of the input genome may be interpreted as an input data quality factor that influences the choice of the most reliable fragment assembly concrete program. For the same class, contigs size (i.e., the number of bases) may be used as an output data quality factor.

Several programs use a sequence database as input. In this case, it is possible to use data consistency or data redundancy as a quality factor. For example, SWISSPROT is a *curated* database, meaning that data consistency is guaranteed, whereas TrEmbl is not curated, that is, it does not guarantee data consistency.

4.3 Workflow optimization

Experience has shown that, in the context of biosequence processing systems, workflows are not too complex and, therefore, they are not amenable to complex optimization procedures, such as those found in current object-relational database systems. Yet, several simple optimization procedures can be profitable employed, as discussed in what follows.

Process Pipelining

Suppose that the workflow contains a sequential composition of two programs, denoted $S_1;S_2$, and that the output of S_1 is a set (or list) of biosequences (that must then be passed to S_2). Then, depending on the semantics

of S_1 and S_2, the system may pass to S_2 the sequences output by S_1, as S_1 produces them, without having to wait for S_1 to produce the full set of output sequences.

For example, consider the composition of Glimmer and BLAST, shown in Figure 5. The ORFs resulting from Glimmer may be immediately passed to BLAST. By contrast, the composition of Phred and Phrap cannot be pipelined because Phrap can assemble the fragments only when Phred generates all reads.

Process Parallelization

Suppose now that the workflow contains a parallel composition of two programs, denoted $S_1//S_2$. Then, obviously, their execution may be fully parallelized.

For example, returning to Figure 5, the contigs generated by Phrap are simultaneously passed to Glimmer and to tRNAScan, which will then execute in parallel.

Note that process pipelining can also be used in combination with process parallelization, for example, in a workflow fragment of the form $S_1;(S_2//S_3)$, if S_1 satisfies the conditions for pipelining with respect to S_2 and S_3.

Data Parallelization

Consider the following simple *data parallelization* strategy:

1. Distribute an input sequence set S among different processors, or assume that the input sequence set is already stored in different processors.
2. Apply the same program P, in parallel, to the sequence sets stored in the different processors, obtaining *local results*.
3. Move the local results to a central processor.
4. Combine the local results, in the central processor, using some *combination procedure* C, to obtain the final result.

The strategy would be correct if the final result is the same as applying P to S.

As an example, consider again the scenario of Figure 5. It is viable to distribute a set of contigs among several processors and then run Glimmer in each of them to parallelize ORF discovery. The final result is simply the union of the local results. This is, therefore, an example of a very simple combination procedure.

Running BLAST is similar. We may distribute the input sequence set among different processors and run BLAST to compare a given sequence against the local sequence sets. The combination procedure would in this case be the merge of the local results [50] that BLAST produced, which is slightly more complex.

Alternative data parallelization strategies are naturally possible by mimicking familiar distributed database query processing optimization approaches [64]. For example, on a first approximation, it is not feasible to run Phrap in parallel to assemble fragments, since all reads must be processed together to ensure correct assembly. A possible strategy to investigate would be to run, in parallel, fragment assembly in stages, that process larger and larger fragments at each stage, until no new fragment is found.

To conclude, there is an interesting case of data parallelization that is worth briefly mentioning. We first observe that there are innumerous concrete programs distributed over the Web that implement the same service, that is, that implement the same abstract program, in out terms. Then, the system may use some form of service discovery, service selection and service composition to achieve data parallelization. Service selection would again be based on quality parameters, such as availability, and service composition would be based on Web service composition protocols or on Grid protocols [48, 49].

Global Optimization

By global optimization, we mean optimizing together several workflows that work on approximately the same data, at approximately the same time, to reduce the volume of data accesses or data moves across different

processors. This might involve simultaneous access to cached data, or even caching data across different transactions, as implemented in object oriented databases [64].

An example is the multi-ring strategy described in [50] that optimizes the combined execution of several BLAST processes.

5 Comparison with Related Work

This section briefly compares systems with respect to how they manage annotations. Specifically, it concentrates on the following aspects (see Section 2 for the list of requirements):

- how annotations are modeled (Requirement R1);
- how annotations are stored (Requirement R1);
- how annotations are accessed (Requirement R2);
- how annotations are versioned (Requirement R3).

We therefore leave out of the comparison other aspects, such as the programming language used to implement the system, or offered to the users, which are not directly relevant to annotation management.

How annotations are modeled

With respect to the data model, a large number of systems use the (pure) relational model, such as GenDB, GeneQuiz, GGB, Manatee and Pedant, whereas others use an object-oriented model or an object-relational model. We may also distinguish between systems that are based on a structured data model, from those based on a semi-structured data model, such as ours.

Systems that adopt a structured data model have several disadvantages. To begin with, it becomes difficult to model the diversity of annotation schemes that the many different data sources and external applications implement. The database designer might end up with a large number of relations or object classes that are difficult to understand and that, often, have a large number of null columns or null attributes. Lastly, the size of the database schema makes it difficult to track new versions.

By adopting a semi-structured model, it becomes simpler to address the problems pointed out above. Most of the issues reduce to creating a flexible set of attributes that apply to the semi-structured objects. Typically, the semi-structure part of an object is stored as a list of attribute / value pairs, often in XML markup. The main disadvantage of this model is that querying the database may become slow. However, recent DBMSs support indexes on XML data, which improve query performance.

As for the database management system itself, several systems, such as GenDB, GGB, Manatee and Pedant, adopt MySQL, which is free. BioNotes opted for Oracle 9 to take advantage of several special facilities the system offers, such as native XML data support.

How annotations are stored

Systems may use a data warehouse approach, where annotations are stored or persisted locally, or adopt a more relaxed storage strategy. In general, a data warehouse approach lets users adjust or correct imported annotations, as well as add new annotations to imported data, freeing him from the restrictions the external data source might impose.

This is best discussed by distinguishing between: annotations imported from external sources; annotations automatically generated by analysis programs; or annotations manually created by users.

Annotations imported from external data sources

Some systems, such as BioNotes and Pedant, store the annotations extracted from public data sources in a data warehouse. Therefore, annotations will be easy to access, but they might not be up-to-date. By contrast,

there are systems, such as MAGPIE, which provide links to the annotations stored in public data sources. Hence, annotations will always be up-to-date, but the link might not be working.

Annotations automatically generated by analysis programs

Some systems, such as BioNotes and GeneQuiz, may pre-execute analysis programs and store the resulting annotations in a data warehouse. This approach avoids waiting for complex analysis programs to finish execution, but it requires re-executing them to maintain the data warehouse up-to-date.

Other systems, such as Imagene, execute analysis programs only on demand. This strategy is easier to implement, but the user will have to wait until the analysis programs end execution.

Annotations manually created by users

Annotation systems also differ on the way they deal with manual annotations. Some systems, such as BioNotes, Artemis, Manatee and Pedant, allow users to store manual annotations in the data warehouse. BioNotes also supports restricted vocabularies, which contributes to improving the quality of manual annotations.

How annotations are accessed

Easy access to the annotations naturally speeds up the discovery of new biological information.

In general, systems may feature a fully centralized architecture, such as Artemis, a "fat" client-server architecture, a "thin" client-server architecture - usually browser-based - such as BioNotes and Pedant, a tightly integrated distributed architecture, a loosely integrated distributed (federated) architecture, or even a Web services architecture.

Systems may also be classified as single-user or multi-user. A single-user system is typically installed on the researcher's workstation and permits only isolated work. By contrast, a multi-user system is typically installed on a server and permits distributed access by a large user community, perhaps through thin clients, which enormously facilitates sharing annotations.

How annotations are versioned

While a genome project is in progress, sequencing of the genome will generate new reads. As a consequence, new contigs are built and, hence, new ORFs, tRNAs and ribosome sites, among others. Hence, it will be necessary to re-execute analysis programs to generate new versions of the data.

Therefore, the system must offer tools to help users compare different versions of the data and transfer annotations from older to newer data versions. However, this is challenging since it is not easy to detect which genome objects (chromosomes, contigs, reads, ORFs, etc) from different versions of the data are identical.

BioNotes and Pedant are prepared to deal with different versions of annotations. The strategies BioNotes adopts are further discussed in Section 6.

6 Conclusions and directions for future research

To give the appropriate context, we began this paper with a brief summary of selected tools and techniques used in Bioinformatics. Then, we listed functional requirements for biosequence annotation systems. We proceeded to describe the functionality and the data model of BioNotes, a tool that meets these requirements. Next, we outlined how the tool will support workflow-like composition of analysis programs. Finally, we compared BioNotes with other annotation systems.

BioNotes is under development at the Catholic University of Rio de Janeiro and in use at Oswaldo Cruz

Institute and at RioGene.

Several issues remain to be addressed in the context of the BioNotes project. The current implementation does not fully support workflow-like composition of analysis programs, which has to be manually programmed. Extending the system to complete cover this functionality is one of the top priorities of the project. Adding new analysis programs is also under consideration, specially those that help mining genome data that hint at new biological knowledge.

We conclude with an issue already discussed in Section 3.1. While a genome project is in progress, sequencing of the genome will generate new *reads,* creating new versions of the data. Hence, the system must offer tools to help users transfer manual annotations from older to newer versions of the data, as otherwise it would be very discouraging to annotate data from on-going sequencing projects. However, this is a challenge since it is not simple to detect which genome objects (chromosomes, *contigs*, *reads*, ORFs, etc), from different versions of the data, are identical.

We are currently experimenting with different strategies to address this problem. Consider *contigs*, for example. One strategy is to consider that two *contigs* are identical if most of their *reads* are the same. Another strategy is to treat two *contigs* as identical if they are best matches when BLASTing *contigs* from the different versions of the data. Yet another strategy is to obtain ORFs using Glimmer, with *contigs* generated from two different versions of the sequencing, and execute BLAST to compare all ORFs (from the old version) with all ORFs (from the new version) in order to detect identical ORFs. After that, the system can transfer manual annotations from the old to the new version. That is, the system can transfer manual annotations assigned to an ORF *f* of the old version to an ORF *g* of the new version, if the system detects that *g* is identical to *f*, using BLAST.

By developing these strategies, we will enable BioNotes to detect identical genome objects from different versions of the data and, hence, transfer annotations from one object to the other.

Acknowledgments

We would like to thank Antonio Basílio de Miranda, Marcelo Alves, Wim Degrave, Paulo Ferreira, Orlando Martins and Marcelo Bertalan for the many discussions during the development of BioNotes. This work was partially supported by CNPq under grant no. 141938/2000-5 for Melissa Lemos.

References

[1] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler, GenBank, Nucleic Acids Research, 31 (2003), 23-27.

[2] A.L. Delcher, D. Harmon, S. Kasif, O. White, and S.L. Salzberg, Improved microbial gene identification with GLIMMER, Nucleic Acids Research, 27 (23) (1999), 4636-4641.

[3] T.M. Lowe, and S.R. Eddy, tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence, Nucleic Acids Research, 25 (1997), 955-964.

[4] Department of Biochemistry and Molecular Biology, Oswaldo Cruz Institute, March 2003, http://www.dbbm.fiocruz.br.

[5] Department of Medical Biochemistry, Federal University of Rio de Janeiro, March 2003, http://www.bioqmed.ufrj.br/.

[6] K. Rutherford, J. Parkhill, J. Crook, T. Horsnell, P. Rice, M-A. Rajandream and B. Barrell, Artemis: sequence visualisation and annotation, Bioinformatics, 16 (10) (2000), 944-945.

[7] S. Pearson, Distributed Annotation System, March 2003, http://www.biodas.org/.

[8] Celera, CeleraBrowser, March 2003, http://www.celera.com/ genomics/commercial/home.cfm?ppage=literature.

[9] S. Moller, U. Leser, W. Fleischmann, and R. Apweiler, EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation, Bioinformatics, 15 (1999), 219-227.

[10] M.G. Reese, G. Hartzell, N.L. Harris, U. Ohler, J.F. Abril and S.E. Lewis, Genome Annotation Assesment in Drosophila melanogaster, Genome Research, 10 (4) (2000), 483-501.

[11] Bioinformatics Group, Center for Genome Research at Bielefeld University, GENDB, March 2003, http://gendb.Genetik.Uni-Bielefeld.DE/.

[12] C. Lee, and K. Irizarry, The GeneMine System for genome/proteome annotation and collaborative data mining, IBM Systems Journal, 40 (2) (2001), 592-603.
[13] S. Hoersch, C. Leroy, N.P. Brown, M.A. Andrade and C. Sander, The GeneQuiz Web server: protein functional analysis through the Web, Trends in Biochemical Sciences, 25 (2000), 33-35.
[14] Berkeley Drosophila Genome Project and The Sanger Institute in Cambridge, UK, Apollo Genome Annotation and Curation Tool, March 2003, http://www.fruitfly.org/annot/apollo/.
[15] Generic Model Organism Project, GBROWSER, March 2003, http://gmod.sourceforge.net/.
[16] C. Medigue, F. Rechenmann, A. Danchin, A. Viari, Imagene: an integrated computer environment for sequence annotation and analysis, Bioinformatics, 15 (1999) , 2-15.
[17] T. Gaasterland, C.W. Sensen, MAGPIE: Automated Genome Interpretation, Trends in Genetics, 12 (1996), 76–78.
[18] Bioinformatics department at The Institute for Genomic Research, Manatee, March 2003, http://manatee.sourceforge.net/.
[19] D. Frishman, K. Albermann, J. Hani, K. Heumann, A. Metanomski, A. Zollner and HW. Mewes, Functional and structural genomics using PEDANT, Bioinformatics, 17 (2001), 44-57.
[20] Rational Genomics, Visual Genome, March 2003, http://www.rationalgenomics.com/visualgenome.html.
[21] University of Washington Genome Center and PathoGenesis Corporation, *Pseudomonas aeruginosa* community annotation project, March 2003, http://www.cmdr.ubc.ca/bobh/PAAP.html.
[22] Center for Genome Research, Whitehead Institute, Community Annotation Project, March 2003, http://www-genome.wi.mit.edu/annotation/microbes/methanosarcina/sarcinaCAP/.
[23] B. Modrek, and C. Lee, Alternative Splicing Annotation Project, March 2003, http://www.bioinformatics.ucla.edu/HASDB/generic.php3.
[24] Bioinformatics Unit, Institut de Génétique Humaine, Montpellier France, GeneStream, March 2003, http://xylian.igh.cnrs.fr/getseq/genbank_sequence_finder.html.
[25] Bioinformatics Laboratory, Institute of Computing - University of Campinas, Cancer Annotation Project, March 2003, http://cancer.lbi.ic.unicamp.br/.
[26] M. Clamp, D. Andrews, D. Barker, P. Bevan, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, T. Hubbard, A. Kasprzyk, D. Keefe1, H. Lehvaslaiho, V. Iyer, C. Melsopp, E. Mongin, R. Pettett, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik and E. Birney, Ensembl 2002: accommodating comparative genomics, Nucleic Acids Research, 31 (1) (2003), 38-42.
[27] R. Agarwala, S. Chetvernin, V. Choi, D. Church, W. Jang, J. Kans, P. Kitts, D. Lipman, D. Maglott, J. Ostell, K. Pruitt, G. Resenchuk, G. Schuler, S. Sherry, T. Tatusova, D. Thierry-Mieg, J. Thierry-Mieg and S. Wheelan, NCBI's Genome Annotation project – current status, March 2003, http://hgm2001.hgu.mrc.ac.uk/Abstracts/ Publish/Workshops/Workshop09/hgm0074.htm.
[28] L. F. B. Seibel, S. Lifschitz, A Genome Database Framework, DEXA (2001), 319-329.
[29] C. H. Wu, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z. Hu, R. S. Ledley, K. C. Lewis, HW. Mewes, B. C. Orcutt, B. E. Suzek, A. Tsugita, C. R. Vinayaka, LS. L. Yeh, J. Zhang, and W. C. Barker, The Protein Information Resource: an integrated public resource of functional annotation of proteins, Nucleic Acids Research, 30 (2002), 35-37.
[30] B. Boeckmann, A. Bairoch, R. Apweiler, MC. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider, The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003, Nucleic Acids Research, 31 (2003), 365-370.
[31] L. Falquet, M. Pagni, P. Bucher, N. Hulo, C. J. A. Sigrist, K. Hofmann, and A. Bairoch, The PROSITE database, its status in 2002, Nucleic Acids Research, 30 (2002), 235-238.
[32] N. J. Mulder, R. Apweiler, T. K. Attwood, A. Bairoch, D. Barrell, A. Bateman, D. Binns, M. Biswas, P. Bradley, P. Bork, P. Bucher, R. R. Copley, E. Courcelle, U. Das, R. Durbin, L. Falquet, W. Fleischmann, S. Griffiths-Jones, D. Haft, N. Harte, N. Hulo, D. Kahn, A. Kanapin, M. Krestyaninova, R. Lopez, I. Letunic, D. Lonsdale, V. Silventoinen, S. E. Orchard, M. Pagni, D. Peyruc, C. P. Ponting, J. D. Selengut, F. Servant, C. J. A. Sigrist, R.Vaughan, and E. M. Zdobnov, The InterPro Database, 2003 brings increased coverage and new features, Nucleic Acids Research, 31 (2003), 315-318.
[33] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, A basic local alignment search tool, Journal of Molecular Biology, 215 (1990), 403-410.
[34] X. Huang and A. Madan, CAP3: A DNA sequence assembly program, Genome Research, 9 (1999), 868-877.
[35] B. Ewing, L. Hillier, M.C. Wendl, and P. Green, Base-Calling of Automated Sequencer Traces using Phred. I. Accuracy Assessment, Genome Research, 8 (1998),175-185.

[36] B. Ewing and P. Green, Base-Calling of Automated Sequencer Traces using Phred. II. Error Probabilities, Genome Research, 8 (1998), 186-194.
[37] P. Green, Documentation for Phrap, March 2003, http://bozeman.mbt.washington.edu/phraps.docs/phrap.html.
[38] Rice P., Longden I., Bleasby A., EMBOSS: the European Molecular Biology Open Software Suite. Trends in Genetics,16 (6) (2000),276-7.
[39] The Institute for Genomic Research, RBSFinder, March 2003, http://www.tigr.org/software/.
[40] The Institute for Genomic Research, TransTerm, March 2003, http://www.tigr.org/software/transterm.html.
[41] National Center of Biotechnology Information, NCBI Homepage, May 2003, http://www.ncbi.nlm.nih.gov/.
[42] National Center of Biotechnology Information, The BLAST Databases, May 2003, ftp://ftp.ncbi.nih.gov/blast/db/.
[43] Workflow Management Coalition (WfMC), Workflow Management Coalition Terminology and Glossar, Technical Report WFMC-TC-1011 3.0. Brussels, 1999.
[44] Workflow Management Coalition (WfMC), The Workflow Reference Model, Document Number TC00-1003.Document Status - Issue 1.1, 1995. http://www.wfmc.org/standards/docs/tc003v11.pdf.
[45] The Workflow Handbook 2003. Fischer , L. (ed.), Lighthouse Point, USA (published in association with the Workflow Management Coalition - WfMC), 2003.
[46] Workflow Management Coalition (WfMC), Workflow Process Definition Interface -- XML Process Definition Language, Document Number WFMC-TC-1025, 2002. http://xml.coverpages.org/XPDLv10.pdf.
[47] N. Guarino, Formal Ontology and Information Systems, Formal Ontology in Information Systems, (Ed. Amsterdam, Netherlands: IOS Press, 1998).
[48] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid - Enabling Scalable Virtual Organizations, November 2003, http://www.globus.org/research/papers/anatomy.pdf.
[49] IBM Corp., The Era of Grid Computing: A new standard for successful IT strategies, November 2003, http://www-1.ibm.com/grid/pdf/it_exec_brief.pdf.
[50] M.A. Casanova and M. Lemos, Optimized Buffer Management for Sequence Comparison in Molecular Biology Databases, Editor C.J.P.Lucena, Technical Report nº 01/01, Computer Science Department, the Catholic University of Rio de Janeiro, 2001.
[51] J.D.Thompson, D.G. Higgins, and T.J. Gibson, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice, Nucleic Acids Research, 22 (1994), 4673-4680.
[52] F. Corpet, Multiple sequence alignment with hierarchical clustering, Nucleic Acids Research, 16 (22) (1988), 10881-10890.
[53] T.F. Smith and M.S. Waterman, J.Mol.Biol, 147 (1981), 195-197.
[54] W. R. Pearson and D. J. Lipman, Improved Tools for Biological Sequence Comparison, PNAS 85 (1988), 2444-2448.
[55] Sun Microsystems. Model-View-Controller, March 2003, http://java.sun.com/blueprints/patterns/MVC.html.
[56] Oracle9 i Application Server Application Developer's Guide, January 2002, http://otn.oracle.com/index.html.
[57] D.K.Casey, Human Genome Program, U.S. Department of Energy, Genomics and Its Impact on Science and Society: A 2003 Primer, November 2003, http://www.ornl.gov/sci/techresources/Human_Genome/publicat/primer/index.shtml.
[58] M. Pop, S.L. Salzberg, M. Shumway, Genome Sequence Assembly: Algorithms and Issues, Computer, 35 (7) (2002), 47-54.
[59] J.Kim, Computers are from Mars, Organisms are from Venus, Computer, 35 (7) (2002), 25-32.
[60] S.F.Altschul, Fundamentals of Database Searching, Trends Guide to Bioinformatics, Elsevier Science, (1998), 7-9.
[61] PHYLIP, November 2003, http://evolution.genetics.washington.edu/phylip.html
[62] PAUP, November 2003, http://paup.csit.fsu.edu/.
[63] TRIPOS, November 2003, http://www.tripos.com/sciTech/inSilicoDisc/moleculeModeling/index.html
[64] M.T. Ozsu, U.Dayal, P. Valduriez, editors. Distributed Object Management. Morgan Kaufmann, San Mateo, 1994.
[65] The Sanger Institute: Informatics Analysis Software: Alfresco, November 2003, http://www.sanger.ac.uk/Software/Alfresco/.
[66] BiBiServ - Bielefeld University Bioinformatics Server - MGA (Multiple Genome Aligner), November 2003, http://bibiserv.techfak.uni-bielefeld.de/mga/
[67] B.Brejová, C.DiMarco, T.Vinar, S.R.Hidalgo, G.Hoguin, C.Patten. Project Report for CS798g, University of Waterloo, 2000. http://citeseer.nj.nec.com/brejova00finding.html.
[68] I. Rigoutsos, A.Floratos. Combinatorial Pattern Discovery in Biological Sequences: The TEIRESIAS Algorithm. Bioinformatics, 14(1):55-67. Errata apareceu em Bioinformatics, 14(2):229, 1998.

[69] I. Jonassen. Efficient Discovery of Conserved Patterns using a Pattern Graph. Technical Report118, Department of Informatics, University of Bergen, Norway, 1996.
[70] S. Henikoff, J. G. Henikoff, W. J. Alford & S. Pietrokovski, Automated construction and graphical presentation of protein blocks from unaligned sequences, Gene-COMBIS, Gene 163 (1995) GC 17-26.
[71] Henikoff, J.G., Greene, E.A., Pietrokovski, S., Henikoff, S., Increased coverage of protein families with the blocks database servers, Nucleic Acids Research, 28 (2000),228-230.
[72] Attwood, T.K., Bradley, P., Flower, D.R., Gaulton, A., Maudling, N., Mitchell, A.L., Moulton, G., Nordle, A., Paine, K., Taylor, P., Uddin, A. & Zygouri, C., PRINTS and its automatic supplement, prePRINTS, Nucleic Acids Research, 31(1) (2003), 400-402.
[73] Bateman, A., Birney, E., Cerruti, L., Durbin, R., Etwiller, L., Eddy, S.R., Griffiths-Jones, S., Howe, K.L., Marshall, M., Sonnhammer, E.L., The Pfam Protein Families Database, Nucleic Acids Research 30(1) (2002):276-280.
[74] Servant F, Bru C, Carrère S, Courcelle E, Gouzy J, Peyruc D, Kahn D, ProDom: Automated clustering of homologous domains, Briefings in Bioinformatics, 3 (3) (2002),246-251.
[75] M. Kanehisa, Databases of Biological Information, Trends Guide to Bioinformatics, Elsevier Science, (1998), 24-26.
[76] Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, Version 1.1: Reference Description, June 2003, http://dublincore.org/documents/dces/.
[77] National Center for Biotechnology Information - 2. Open Reading Frame Finder, November 2003, http://www.ncbi.nlm.nih.gov/gorf/gorf.html.