



GAIA: Framework Annotation of Genomic Sequence

L. Charles Bailey, Jr., Stephen Fischer, Jonathan Schug, et al.

Genome Res. 1998 8: 234-250

Access the most recent version at doi:[10.1101/gr.8.3.234](https://doi.org/10.1101/gr.8.3.234)

References This article cites 21 articles, 3 of which can be accessed free at:
<http://genome.cshlp.org/content/8/3/234.full.html#ref-list-1>

License

Email Alerting Service Receive free email alerts when new articles cite this article - sign up in the box at the top right corner of the article or [click here](#).



To subscribe to *Genome Research* go to:
<https://genome.cshlp.org/subscriptions>

Cold Spring Harbor Laboratory Press

RESEARCH

GAIA: Framework Annotation of Genomic Sequence

L. Charles Bailey, Jr.,¹ Stephen Fischer, Jonathan Schug,
Jonathan Crabtree, Mark Gibson, and G. Christian Overton

Computational Biology and Informatics Laboratory, Department of Genetics, University of Pennsylvania
School of Medicine, Philadelphia, Pennsylvania 19104-6021 USA

As increasing amounts of genomic sequence from many organisms become available, and as DNA sequences become a primary reagent in biologic investigations, the role of annotation as a prospective guide for laboratory experiments will expand rapidly. Here we describe a process of high-throughput, reliable annotation, called framework annotation, which is designed to provide a foundation for initial biologic characterization of previously unexamined sequence. To examine this concept in practice, we have constructed Genome Annotation and Information Analysis (GAIA), a prototype software architecture that implements several elements important for framework annotation. The center of GAIA consists of an annotation database and the associated data management subsystem that forms the software bus along which other components communicate. The schema for this database defines three principal concepts: (1) Entries, consisting of sequence and associated historical data; (2) Features, comprising information of biologic interest; and (3) Experiments, describing the evidence that supports Features. The database permits tracking of annotation results over time, as well as assessment of the reliability of particular results. New framework annotation is produced by CARTA, a set of autonomous sensors that perform automatic analyses and assert results into the annotation database. These results are available via a Web-based query interface that uses graphical Java applets as well as text-based HTML pages to display data at different levels of resolution and permit interactive exploration of annotation. We present results for initial application of framework annotation to a set of test sequences, demonstrating its effectiveness in providing a starting point for biologic investigation, and discuss ways in which the current prototype can be improved. The prototype is available for public use and comment at <http://www.cbil.upenn.edu/gaia>.

As large-scale genomic sequencing becomes more common, the contribution of DNA sequence data to gene identification and characterization is changing rapidly. Until recently, other techniques were used to narrowly define a region of interest or to isolate a candidate cDNA for a particular phenotype. Sequencing has generally been undertaken at a late stage in the experimental process, and been restricted to cDNAs or short genomic regions of interest (e.g., translocation breakpoints) because it is an expensive and relatively labor-intensive process in most laboratories. The growth of sequencing centers and core facilities in larger institutions has improved access in many cases, but sequencing has been considered cost-effective only when the very fine structural resolution it provides was necessary to reach a biologic conclusion. Until recently, therefore, most sequence data has been directly con-

nected to transcribed regions of DNA, and has been generated only after compilation of functional information about the region, which has guided the interpretation of the sequence data.

Genome-scale sequencing imposes new conditions of quantity and context on the problem of understanding sequence data. As the name "genome-scale" implies, the goal is to make readily available sequence for at least one allele of any region from a species' genome, so that it can be used as a primary reagent when investigating a locus of interest. However, clones are usually selected for sequencing based on mapping data only. The resulting sequence will generally not have significant functional information associated with it, nor will it be the role of the sequencing centers to perform further investigation; this will continue to be the province of laboratory groups with an interest in a particular biologic problem.

For the sequence to be most useful to these groups, though, it will be necessary to associate with

¹Corresponding author.
E-MAIL bailey@www.cbil.upenn.edu; FAX (215) 573-3111.

it a framework of biologic information that can be used to identify regions of particular interest, and to guide investment of laboratory resources in studying these regions. It was initially suggested that this process of annotation was best left to individual investigators, who could apply their particular expertise to tailor the analysis of sequence that was of interest to them. There is growing agreement, however, that a systematic effort should be undertaken to generate an initial set of annotation. This permits the application of specialized resources to the process, and gives investigators more rapid access to the information that is most useful to them. Such information includes both functional data, such as content of genes or motifs important in determining chromatin organization, and structural data, such as locations of mapped markers or repeat content. The principal goal of this process, which we call framework annotation, is not to describe completely the biologic characteristics of given sequence, but to provide to later investigators sufficient information to orient themselves and determine how best to focus their efforts.

Framework Annotation

There are several characteristics that the framework annotation process should possess. Because the current goals for human genome sequencing require that ~2 Mb of new sequence be produced each day, the mechanisms used for annotation must be capable of sustaining a high sequence throughput. This implies that computational techniques will be an important source of data, because they are less labor-intensive and scale up more easily than most laboratory methods. In some cases, computational methods are already definitive (e.g., for repeat identification) or nearly as reliable as laboratory experiments (e.g., locating STSs). Gene-related annotation is more error-prone, but reasonably accurate initial predictions can be obtained through careful use of both pattern-matching algorithms such as GenScan (Burge and Karlin 1997) and GRAIL (Uberbacher et al. 1996), and increasingly through similarity search, especially with EST databases as targets (Bailey et al. 1998).

If framework annotation is to serve as a guide for development of further experiments, the data it provides must be reliable enough that resources will not be invested frequently in pursuing false leads. In many cases, this can be done by considering only the best results from a particular technique. This runs somewhat counter to the approach taken by most current tools, which tend to emphasize sensi-

tivity at the expense of specificity. It is an important consideration for high-throughput annotation, though, in which the extensive human filtering and editing needed to interpret the complete output of many current programs is not practical. In some cases, reliability of data can also be improved by combining information from multiple sources, especially ones based on complementary theoretical grounds. Each type of annotation should also be as complete as possible, so that users have access to as much information as it can reliably produce, and can also draw conclusions based on the absence of annotation, as appropriate to a method's sensitivity.

It is also important to maintain currency of annotation. In some cases, results will be fairly stable (absent changes in the annotated sequence), so the problem is primarily one of adding new data as it becomes available (e.g., similarity to newly described transcripts). In general, though, provision must be made for changes to existing data, whether because some underlying information changes naturally over time (e.g., size of target databases, and therefore probabilities estimating significance of similarity) or because the analytical methods are revised. We describe this problem using the term "annotation fade and refresh." The fundamental notion is that different pieces of annotation become obsolete, or fade, at different rates and must be refreshed by periodic reanalysis. If compatible annotation is not regenerated, a datum is considered less reliable, and eventually may be removed entirely. This process requires an understanding of how well annotation produced by a given method holds up over time, and what constitutes a compatible result from a later analysis, which may in some cases be very difficult to determine.

It will be essential to clearly indicate the source of data for all results, so that users can select information useful to them and assess its validity in their specific circumstances. The feature description language (FDL) used in the widely available IC databases (DDBJ, EMBL, and GenBank) provides only limited means for expressing this sort of information in structured form (DDBJ/EMBL/GenBank 1996), and these are often not used when sequences are submitted. The Genome Sequence Database (GSDB) has addressed this problem in greater depth in its current schema (NCGR 1996), although it still depends on data depositors to use these optional fields. These facilities are consistent with the environment in which the databases developed: Most data were based on some type of laboratory evidence, and users wishing to assess its validity in detail did so primarily by consulting literature refer-

BAILEY ET AL.

ences associated with the entry. This model does not extend well to genome-scale sequencing, though, in which there is often limited connection between a given sequence and the literature, and the large volume of new sequence makes describing the data only in external literature references impractical in any case. It will be important, then, for framework annotation to carry with it some record of its origin, though many methodologic details can be abstracted into a common reference point, as long as it is readily available.

In addition to representing underlying biology as accurately as possible, framework annotation should be recorded consistently across different instances of data (e.g., different features of a given type) and different sequences. This can be done in part through structured representation of the data, which imposes a well-defined form within which values vary from instance to instance. In addition, semantics of specific fields may be further defined by use of standardized terms and controlled vocabularies. This approach simplifies the process of comparing different instances of data and allows users to generalize experience from case to case. More importantly, it makes it possible to construct formal queries over the contents of records, and not just their existence. This is useful not only for retrieving a subset of annotation from a specific locus, but also for assembling data sets in large-scale studies, such as assessment of the annotation engines or preliminary investigations of genome-wide structure.

Given these requirements for throughput and content, we expect that automated processes will contribute the majority of data to framework annotation. Properly designed automated tools can produce consistent annotation over time, structured according to well-defined process models. Defining these models is, of course, the most difficult aspect of automation. In part, this hinges on recognizing the strengths of different analytical methods to filter data properly. It also requires careful definition of data structures, so that large amounts of information may be rapidly queried. Finally, methods will require careful testing on well-defined benchmark sequences and regular reevaluation as reference genome sequences become more complete. This process has already been undertaken in connection with sequencing of prokaryote (Tomb et al. 1997), yeast (Ouzounis et al. 1995), and *Caenorhabditis elegans* (P. Green and L. Hillier, in prep.) genomes. Although much can be learned from this experience, we expect that annotation of mammalian genomes will be more difficult because of their larger size and more complex gene structure.

New mechanisms will also be needed to make framework annotation available to users. Established text-based methods are strained by the volume of data present over large sequences. This has led to development of several graphical visualization interfaces (Searls 1995; Helt 1996; Harris 1997) designed to present a dense view of data in an intuitive fashion. Within such images, the tradeoff for density is often a lack of the fine detail supplied by textual displays. This can be addressed by providing context-sensitive cues in the display, or linking together multiple simultaneous representations of selected data. However, the integration of multiple types of data and different levels of detail into a widely available user interface remains a challenge for the computational genomics community.

In addition to displaying static data, visualization tools may need to generate dynamic information at the time of display, where this proves better than archiving of some metric (e.g., GC content). They should also include mechanisms through which users can apply additional analyses to selected regions. This is particularly important for problems lacking a definitive analytical technique (e.g., pattern-matching gene finders); whereas trying to present data from all of the available alternatives at once may produce a cluttered display, users may want to perform additional analyses in specific cases in which they may function especially well.

Provisions will also need to be made for allowing users to directly contribute annotation by adding new information or editing existing data. This is part of the broader requirement that genome-scale annotation must be able to accommodate changing information over time, so that public databases serve less as an archive of completed annotation and more as a representation of developing understanding. This will again be a difficult process, as it will require maintaining reliability and consistency across input from multiple individuals, as well as tracking the origin of data and maintaining its integrity. The public databases GSDB and Genome DataBase (GDB) have each taken initial steps in this direction within their respective domains. Much work remains to be done though, both in the development of community standards for annotation, and in the practical mechanics of data exchange among existing databases and analytical tools. For these reasons, we expect that real progress in this area will be slower than in the process of generating automated framework annotation.

A number of systems for sequence annotation are under development; an excellent overview can be found in Harris (1997). At the lowest level, well-established analysis packages such as Genetic Data

Environment (GDE) (Smith et al. 1994), and the GCG suite (Genetics Computer Group 1996) of which GDE is now a part, have for many years provided a common user environment for execution of many analyses, without really providing a model for annotation per se. More recently, tools such as PowerBlast (Zhang and Madden 1997) have added a basic notion of annotation to their analytical foundation, allowing display and export of results as features along a sequence. Database submission tools (Harger et al. 1997; Wolfsberg et al. 1997) have begun to provide facilities for checking annotation at the time an entry is composed, and, in the case of GSDB, for explicitly recording computational annotation. More comprehensive systems are becoming established for simpler organisms: MAGPIE (Charlebois et al. 1996) is currently in use for prokaryotic genomes, and GeneQuiz (Scharf et al. 1994) has been used to predict functions for open reading frames (ORFs) in *Saccharomyces cerevisiae*. Because the human sequencing effort is just reaching its stride, and other mammalian genome projects are at earlier stages, such systems are less well established for annotation of sequence from higher eukaryotes. Nonetheless, prototypes do exist, and initial efforts are under way to make the transition from individual projects to a wider scope. The annotation consortium based at the Oak Ridge National Laboratory is continuing work on the Genome Channel (Uberbacher et al. 1997), designed to collect both sequence and results from multiple analyses and provide a window into human genome sequencing. Free-standing packages designed to support large-scale annotation exist as well, such as the freely available Genotator (Harris 1997) or the commercial GeneWorld (Pangea Systems 1997). The principal focus thus far has been on generating data, presenting it to the user, and accepting user input; less emphasis has been placed on development of structured database schemata, consistency of data represen-

tations, or data exchange. Nonetheless, these systems provide significant new capability for annotation of large sequences, and will undoubtedly contribute heavily to the development of future annotation systems.

The GAIA Architecture

Here we describe a prototype software architecture, GAIA (Genome Annotation and Information Analysis), designed specifically to address questions raised by the process of genome-scale framework annotation (Fig. 1). It is intended to provide a structure within which initial approaches to framework annotation may be implemented and tested, data may be tracked to assess the annotation process over time, and interfaces may be developed to provide better visualization and retrieval of data. Although the prototype does generate annotation of several types, we expect that its greatest long-term utility will be less as a definitive collection of methods than as a testbed in which the annotation process

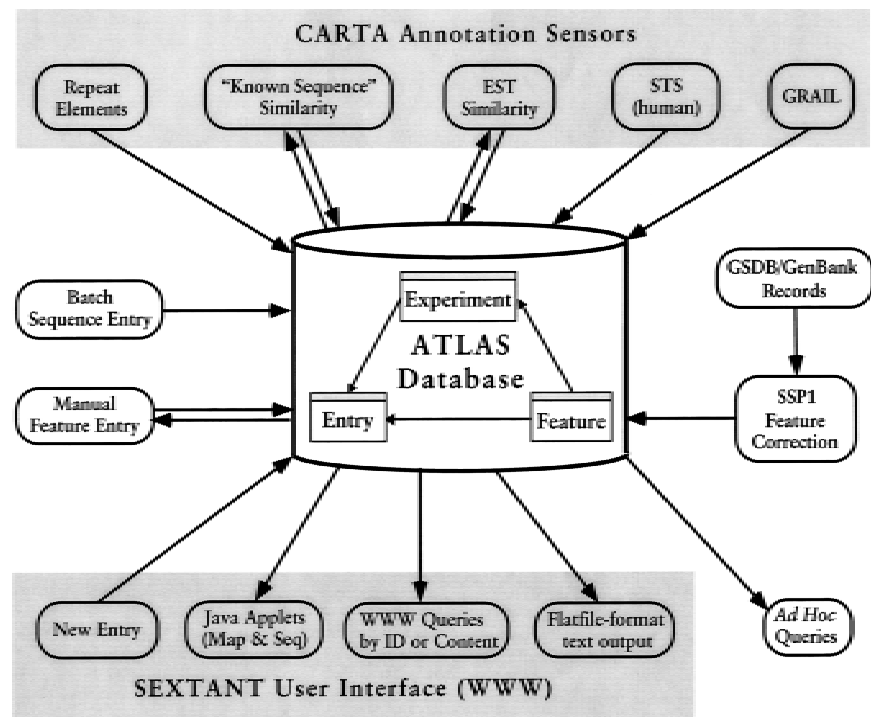


Figure 1 Overview of the GAIA architecture. Components of each subsystem are grouped together. Arrows directed toward the database indicate flow of new information into ATLAS; those directed away denote retrieval of data. External sources of data (e.g., dbEST) used by some of the CARTA sensors are not shown. Within the database, the relationships among the three principal classes of data are shown; the arrows indicate presence of a reference in an object to a member of another class. Individual components are explained in greater detail in the text.

BAILEY ET AL.

may be examined carefully, with new approaches incorporated as their value is demonstrated.

We have constructed GAIA as a set of autonomous components interconnected along a software bus, rather than a single highly integrated system. This facilitates addition or revision of components, as well as examination of individual methods where appropriate. We describe these components in greater detail below.

ATLAS

At the center of GAIA is the data management system (ATLAS), which functions as both an archive and the bus along which other components communicate. This provides a common data exchange API, and carries with it a number of useful properties: production of new annotation by each sensor may be tracked directly; because communication occurs via ATLAS, specialized channels do not need to be developed for different combinations of sensors; each component is insulated from the application-specific data formats of other components; and the process of adding or removing analysis or display components is simplified.

In our prototype implementation, we have focused on defining a data model and schema, particularly in those areas directly related to automated annotation. As we discuss below, we have needed to make several trade-offs to accommodate the particular database management system (DBMS) used in the current version, but in each case have tried to construct the current schema in a way that will easily accommodate revision to a more formally correct and robust version. Such a revision will be an essential step in developing a public editing interface, because it will need to specify mechanisms for insuring semantic consistency across input from a wide variety of users.

Schema Definition Language

The data exchange format used throughout GAIA is Schema Definition Language (SDL), an extension of the Collection Programming Language (Buneman et al. 1994) developed for database integration. SDL is able to represent the complex data types often needed to describe biologic concepts. In addition, it is not bound to a particular type of DBMS; this makes it possible to modify or replace the database implementation without the need for extensive redesign in the rest of the system.

Annotation Database Schema

We have constructed over this base a set of domain-

specific data types that constitute the schema for the prototype annotation database. It defines three principal classes: Entry, Feature, and Experiment. Each object in one of these classes carries a unique identifier, and is independent of objects in the other classes, although it may refer to them. In addition to biologic data, each object carries an audit trail showing when modifications were made, so that the state of data may be tracked over time. It also carries an access identifier, on which is based a simple mechanism for restricting access to confidential data prior to publication.

We have used the schemata of public sequence databases as a starting point for our definitions, to incorporate their experience in meeting the needs of the biologic community, and to more easily accommodate records originating in those databases. Our current treatment of sequence is therefore entry-based, meaning that the unit of reference for sequences is the individual entry rather than the complete genome. In addition to the sequence itself, the Entry class includes historical information, such as the origin of the DNA (species, map location, etc.), where it was sequenced, descriptive information, identifiers from outside databases, and references. Most of these fields are formally optional for compatibility with public databases, but GAIA's input tools require that most historical data and at least one identifier and reference be specified, so they are effectively mandatory fields for new data. Type definitions for fields such as species of origin will need to be defined further by use of broader controlled vocabularies to insure greater consistency in new data. To provide a more logical view of genome scale sequence, and to best resolve such phenomena as overlapping entries, we intend for this schema to evolve to a chromosome- or genome-based schema, at which time the current entry descriptions will become features associated with part of a larger sequence.

Biologic information is encoded by the Feature class. This includes both functional data, such as gene annotation, and mapping data, such as locations of STSs and repeat elements. The biologic meaning of a Feature object is specified principally by its Type field, with additional data supplied as a list in the Qualifier field; the types of qualifiers used vary across different feature types. We have chosen to specify Feature as a single class in the current schema because different feature types share most elements of their structure; the principal differences occur in the Type values and some of the associated qualifiers. However, this does make it more difficult to define constraints specific to certain feature types

(e.g., required qualifiers). In the current prototype, these constraints are enforced during the generation of features, but better specifications may be obtained in the future by implementing feature type-specific subclasses of Feature. The current set of feature types includes those defined in the IC Feature Description Language, plus several new types reflecting the output of computational analyses.

Each Feature is associated with a particular sequence by a reference to an Entry, and specification of a Location in that entry. In the current schema, the Location type may specify an ordered list of one or more contiguous spans, each of whose endpoints may be discrete, or may be one of a range of values (equivalent to the $\langle a, \rangle b$, and $a.b$ notation of the IC FDL). A more complete Location specification, which is capable of defining alternate paths through sequences, is under development. Because this is a relatively rare occurrence, however, and because there is not yet a compact way to display this type of location in the graphical user interface, we have elected to use the simplified Location in the current system. We have focused our initial work on the feature types enumerated in the complete schema description (principally gene-related features); other feature types currently exist only in records imported from IC database entries. Gene-related features currently also contain a GeneName field, which is used in the current prototype to associate related features. In a schema incorporating feature type-specific classes, this would be better modeled by a containment hierarchy of feature classes (e.g., an mRNA class containing reference to objects in an exon class, from which it would take its location). We have not done this for pragmatic reasons in the current prototype; such reconstruction imposed an unacceptable performance penalty for the current DBMS.

The source of a Feature's data is specified in two ways. Within a Feature object, the Origin field contains a keyword drawn from a controlled vocabulary specifying the general means by which the data were obtained. Current values are *automatic* (feature generated by a CARTA sensor), *manual* (feature generated from data provided to ATLAS by direct human contribution), and *GenBank* (feature transformed from a description in an IC database record). Its primary uses are to indicate the level of review needed, and to identify Features that come from IC databases, and so may contain incorrect usage or data types that need to be more fully specified in future schema revisions.

More importantly, each Feature contains references to one or more Experiments, which specify in greater detail the process by which the data were

derived. This may be a laboratory experiment, a computational experiment, or information directly specified by a human (termed an *author* experiment). As with Feature, the principal semantics are provided by the value in the Type field. Once again, extension of the schema to specify a series of experiment type-specific subclasses would permit better structuring of other fields and better constraint checking; it is not done at present for practical reasons. Details of the Experiment are carried in the Date, Parameters, Result, and Version fields. While the last three are currently unconstrained strings, they contain structured text for all automatically generated Experiments, in anticipation of more structured definitions in future schema revisions. Finally, an Experiment contains a reference to the single Entry supplying the sequence on which the experiment was performed. Although this is generally sufficient, it may be relaxed in the future to permit experiments that consider several sequences (e.g., overlapping portions or allelic variants), particularly as the schema moves away from an entry-based model.

This schema definition extends existing schemata, such as the IC FDL, in several ways. By making Entry, Feature, and Experiment separate, structured, and uniquely identified objects, it simplifies querying on a particular attribute across multiple entries, because each item is identifiable and accessible without indirection through another type of object. This will also facilitate the process of resolving overlapping and incomplete sequences into an integrated view, as sequences and Features can be merged independently. In addition, the history of each object can be tracked over time. The recent addition of feature identifiers in GenBank is a similar step, as it has become generally recognized that features are frequently retrieved in their own right, and as much in need of being current as the sequence itself.

Most important, though, is the addition of the Experiment class. This makes it possible to ascertain in detail on what evidence a given feature is based. It also permits direct queries to determine what other types of evidence might be available for the sequence under study, and what information has been produced by a given method across the database. This makes it possible to perform a critical analysis of not only individual sequences, but also the annotation process itself.

Database Management System

The current ATLAS prototype operates over a rela-

BAILEY ET AL.

tional database, converting to and from SDL as required. For reasons of performance, we have found it necessary to permit direct SQL queries from SEXTANT against fields broken out of the SDL into separate columns in the relational tables. No bypass exists for data entry, however, because each object must exist as SDL to satisfy the schema. All objects are versioned, with obsolete versions maintained in separate tables to improve performance of queries against live data.

Data Entry and Editing

New information may enter the annotation database in several ways. We have written a parser for GenBank flatfile-format entries, which generates corresponding Entry, Feature, and Experiment objects. Before creating Features, the SSP1 parser/corrector (Overton et al. 1994) is used to scan gene-related features; although it cannot handle all cases, it does correct a number of common errors, and infers features that are frequently missing in IC entries. Features arising from GenBank information have the Origin value `GenBank`, and are associated with Experiments of type `author` or `experimental`, in an attempt to reflect the existing support for evidence in the IC FDL.

Entries may also be generated directly for new sequence, by use of either a batch tool that reads flat files, or the SEXTANT entry form. Features may be input directly with a flatfile that defines Feature and Experiment objects by use of a format similar to the GenBank flatfile feature table. The current parser for this format performs very basic consistency checks, but still relies on the user for the majority of quality assurance. Finally, appropriate Feature and Experiment objects are generated directly by CARTA sensors.

Editing of existing data may be performed in one of two ways. A simple text-based tool permits pattern-based direct modification of SDL, with no consistency checks and limited versioning support; this is currently used only to correct minor textual errors. A minimal record-based editing interface is provided by an editor written in PerlTk. This contains better support for syntactic checking and versioning, but it does not provide any semantic constraint checking at present. We are developing a Java-based parser as the foundation for a new editor interface. Although this will improve the editing interface, we expect that it will be difficult to construct a general-purpose editor. The greatest difficulty in this effort is the need for robust semantic as well as syntactic constraint checking to insure con-

sistent usage throughout the database. Postprocessing by tools such as SSP1 and human curation provide some of this capability, but we expect checks at the time of entry to be both more effective and more flexible, as the user can correct the data to reflect his intent directly.

CARTA

The CARTA subsystem is responsible for automated generation of framework annotation. It comprises a set of autonomous components, called sensors, each of which performs a specific analysis, and records its results via ATLAS. A sensor also draws its principal input from ATLAS; this may include the sequence itself, other information about an Entry, and the results produced by prior sensors. Some sensors also make use of outside data (e.g., sequence databases), in which case they record identifying information in their Experiment objects. The actual analysis may occur directly within the sensor's code, or in a separate tool whose output the sensor transforms to fit the annotation schema. Results are submitted to ATLAS as Feature and Experiment records constructed directly by the sensor; in the current prototype, this is the point at which most data consistency checks are performed.

Annotation Policy Statements

For users to assess the reliability of annotation, or to use it effectively in further experiments, it is essential that they understand the process by which it was produced. Although some of this information is encoded in the Experiment objects associated with every GAIA Feature, these descriptions are designed to record the details of a specific experiment. To explain the ground rules under which a particular automated sensor or manual annotation process operates, we have developed a series of publicly available annotation policy statements. Each statement specifies the methods used and criteria applied to produce the results from a particular process, and the form that those results take. These policy statements also help to clearly define a method as it is being developed, and may serve as a basis from which discussion of revisions to the process can proceed. Such dialogue will play a critical role in the informatics community, and it is hoped in the genomics community as well, in developing a coherent approach to genome-scale annotation.

Initial Sensors

The CARTA prototype currently includes five sen-

sors, three of which perform similarity searches, one of which invokes a pattern-matching gene finder, and one of which attempts to place PCR-based STSs on the sequence (Fig. 2). Our intent is to include examples of major analytical methods useful for framework annotation, rather than to construct a comprehensive representation of currently available tools. All of the current sensors except the STS sensor are capable of annotating human, mouse, *Drosophila*, or *Arabidopsis* sequence.

Because our goal was to generate reliable data with minimal human intervention, we have restricted the current framework annotation to high-quality output of well-characterized methods. Although this leads to some loss of sensitivity, we have found that it is difficult to distinguish signal from noise at lower stringencies without human intervention. As the performance of existing methods on large sequences is better characterized, new methods are implemented, and effective ways to combine data from multiple sources are developed, we expect that it will become possible to draw more reliable conclusions from low-stringency data.

The first of the similarity-based sensors is IREsense, which detects interspersed repeat elements (IREs), simple sequence repeats, and cloning vector sequence, using the CENSOR package (Jurka et al. 1996) as the analytical engine and a set of repeat templates based on Repbase (Jurka et al. 1992). Subtyping of *Alu* repeats is done by use of PYTHIA (Milošavljević and Jurka 1993). Results are encoded as

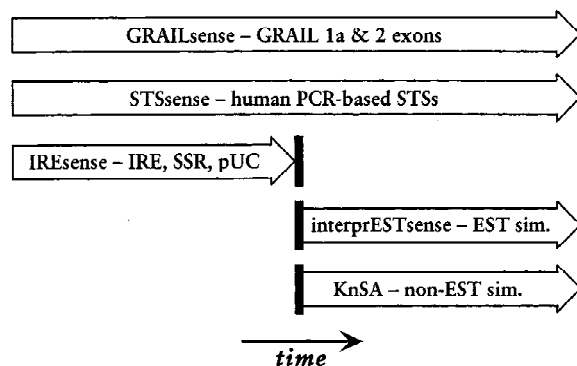


Figure 2 CARTA sensors. The diagram shows the sensors currently active in the CARTA prototype. The horizontal axis represents elapsed time, with the far left being the time at which annotation of a sequence is started. Each sensor may be run as part of an annotation strategy at any time within the corresponding open bar in the diagram. Overlaps between bars show where sensors may be run in parallel; functional dependencies appear as bars that do not overlap.

Features of type `repeat_region`, with qualifiers specifying the type of repeat and the portion of the template matched. These Features are then used by the other similarity-based sensors to generate a masked query sequence. `interprESTsense` uses a BLAST-based strategy to detect high-stringency similarity to expressed sequence tags from any species in dbEST. `KnSA` uses a similar method to detect similarities to non-EST sequences in GenBank. Because the intent is to detect near identical sequences, and the thresholds are set higher than the level of identity seen for most cross-species conservation (Makalowski et al. 1996), taxonomic divisions of GenBank other than the one matching the query sequence are excluded. Similarly, the synthetic (`syn`), genome survey (`gss`), and unfinished high-throughput (`htg`) divisions are excluded, because entries in these divisions are less likely to contribute significant biologic information to the annotation. Each sensor generates Features of type `similarity`, whose qualifiers contain a summary of the alignment scores.

Pattern-based gene finders are represented by `GRAILsense`, which serves as an adapter between GAIA and the GRAIL command-line client. It creates `grail_exon` Features based on `grail1a` and `grail2` exon predictions, recording the originating algorithm, score, and enclosing ORF with each feature. Because the behavior of GRAIL on long genomic sequences was not well understood when we constructed the prototype sensor, we set the threshold for retention at the `good` quality level based on anecdotal experience in a small test set of sequences. We selected GRAIL as CARTA's initial gene finder in part because its accuracy was among the highest when compared with several other algorithms (Bursset and Guigo 1996), in part because it is widely used, and therefore may be familiar to individuals working with GAIA's annotation, and in part because it provided a client interface readily incorporated into a CARTA sensor. Addition of other pattern-based algorithms, especially newer tools such as GenScan, may provide useful information. However, the incremental benefit may be small (Harris 1997), and may come at the cost of a large amount of redundant data. It will therefore be necessary to examine not only single-engine sensors, but ones reporting some combination of results from complementary algorithms. The best approach will also vary for pattern-matching algorithms with different goals; for example, tRNA gene finding is likely to require a significantly different mix than identification of protein-coding genes.

The last of the prototype sensors, `STSsense`,

BAILEY ET AL.

identifies STSs in genomic sequence using PCR primer pairs and predicted product sizes. The current implementation uses chromosome-specific data sets of human STSs, so its function is limited to human sequences whose chromosome of origin is known. Matches are recorded as STS features, with qualifiers specifying the primers and expected product size range, as well as cross-references to the database from which the STS description was drawn.

We have begun to address the issue of updating results from these sensors over time. For *interpretEST*-sense and *KnSA*, the ID of the target EST or GenBank entry supplies a natural means for distinguishing between new and changed features; the update mechanism revises the Feature object for a given target sequence with new results on each run. A similar mechanism is being implemented for *STS*-sense. As STS features do not record any values (such as match probabilities) that vary naturally over time, however, we do not expect these features to change rapidly. This is also true of *repeat_region* features, for which the identification of compatible new results is more difficult, because there is not a unique pairing of each template with the query sequence. Because the features cover contiguous spans, our initial approach is to match results on the basis of overlap and template name. Difficulties arise, however, when new results would require a change in repeat template, or merger of old features. This problem is more pronounced in the case of *GRAIL*-sense, in which we have encountered changes implying merger of more than one existing feature and splitting of a single existing feature, as well as shifting of features along the sequence. As a result, we have not yet developed an effective update strategy for this sensor. Wholesale deletion and reconstruction of *grail_exon* features represents a minimal approach, but we hope to develop a model that better preserves the identity of features across repeated analyses.

As is apparent from this description, the current sensors do not create traditional gene-related features such as exons, mRNAs or 5' or 3' untranslated regions (UTRs). We have found that none of the current sensors produces results accurate enough to reliably predict both the existence of a feature and its boundaries correctly, so we have refrained from describing their results directly with these feature types. Although the explicit links to supporting evidence would distinguish such tentative features in GAIA from those more firmly established, we are reluctant to use descriptors that biologists associate from prior experience with experimentally defined gene structures, particularly if translation into other

databases might make this distinction less apparent. We expect that combination of evidence from multiple first-tier sensors will produce more reliable gene models; it may then be appropriate for higher-level sensors to create gene-related features.

Scheduler

The automated annotation process is controlled by the CARTA scheduler. It invokes individual sensors in accordance with a specified annotation strategy that must take into account functional dependencies between sensors. The prototype scheduler implements a default strategy that executes all of the current sensors, or a user-specified subset, in series for each entry processed. For batch processing of multiple entries, we have used several scheduler processes to run independent sensors in parallel.

SEXTANT

The primary user interface to GAIA is provided by the SEXTANT subsystem, that handles both queries against current data and submission of single new sequence entries. Because we wanted to make this system widely available without requiring specialized software, SEXTANT operates via the World Wide Web by use of a set of HTML forms, text pages, and graphical displays. With the exception of new entry submission, SEXTANT currently provides read-only access to the database, but we expect to develop the editing interface using Java applets supplemented by HTML forms.

Graphical Interface

The graphical portion of SEXTANT comprises a pair of linked Java applets based on the *bioWidgets* (Crabtree et al. 1998) map and sequence viewers (Fig. 3). The map display provides a potentially dense low-resolution view of an entry, presenting information via position, shape, and color in a display similar to gene diagrams familiar to most biologists from the literature. The current map applet displays results of CARTA annotation, as well as several types of gene-related feature. Position above or below the horizontal axis indicates the strand on which a feature appears; its location in the sequence is denoted by offset along the axis. Related features are collected into tiers at different distances from the axis, and specific feature type is indicated by color. In addition to the features themselves, bars are provided that indicate types of evidence supporting

BAILEY ET AL.

gene-related features. The display may be scrolled both horizontally and vertically, and the horizontal scale adjusted to provide the desired level of resolution. Individual spans can be selected by clicking on them. Doing so highlights that span with a dark outline, and other spans in the same feature with a lighter outline. A brief summary of the selected item appears in the text window at the bottom of the display. Groups of features may be hidden on the basis of supporting evidence; a feature is displayed as long as any type of evidence that supports it is selected. Finally, buttons at the bottom of the display call up more detailed textual descriptions of the selected feature or the entire entry in a separate browser window.

The sequence display provides a higher resolution view of the entry, displaying nucleotides as text, with features indicated by lines beneath the sequence. Feature type is indicated both by color coding matching that in the map display, and by a label on each line. Feature location is denoted by the span of the line, with polarity displayed as an arrowhead at the end of the line. The display may be scrolled vertically, or repositioned directly to a specified offset. Features may be selected by clicking on the label; doing so selects both the feature and the range of sequence it spans. Sequence ranges may also be selected directly by clicking and dragging. A conceptual translation and GC content graph may be calculated on the fly and displayed in different formats. Several other controls are provided for modifying display characteristics such as layout, font, and color. In addition, user-specified labels may be added over arbitrary spans; although these remain in the display as long as it is present, they are not transmitted back to ATLAS. The selected range of sequence may be exported for further analysis via a URL specified in the applet's startup parameters. Results are displayed by use of the called service's usual output mechanisms. The current SEXTANT prototype includes a single option of this type: search for transcription factor binding sites

with TESS (Schug and Overton 1997). Finally, provision is made for printing the display by sending a PostScript description of its contents to the browser.

The sequence and map displays are linked, so that selection of a feature in one causes it to be selected in the other as well. In addition, when a feature is selected in the map display, the sequence display scrolls so that it is visible there as well. This provides a coordinated view, and makes the facilities of both applets available for further examination of the selected feature. The region visible in the sequence display is also indicated on the map display to aid in registration.

Text Displays

Data may also be displayed as text to provide more complete descriptions than the graphical interface can present. Individual objects may be retrieved in a format designed to be easy to read, in a format with the SDL record structure more completely noted, or as a pure SDL expression. In the former two cases, references to IDs within ATLAS or in external databases are hypertext links to the owning database's Web display of that object. When a query may return a collection of objects, it is also possible to view results as a summary table, with links to more detailed displays. Sequence may be retrieved as a numbered display for direct reference or a packed block of text for ease of export.

To provide a basic method for data exchange, or to allow a user to save all data about an entry, a GenBank flatfile format display may be obtained for an Entry and all of its associated Features. CARTA-generated feature types that are not part of the IC FDL are represented as `misc_features`, with the type noted in a `/label` qualifier. Additional information is recorded as structured text within `/note` qualifiers. A unique `/label` is generated for every feature, because that is the only mechanism provided by the IC FDL to uniquely identify a feature within an entry. The ATLAS ID for each feature is

Figure 3 SEXTANT display components. The image shows SEXTANT's map (*top right*) and sequence (*bottom*) applets displaying part of a gene-rich region on human chromosome 22. Bars in the map applet indicate the type and location of individual features along the sequence, with gene-related features closest to the axis, and results of CARTA annotation in the outer tiers. The selected feature is marked by a black outline. The gray lines surrounding this feature and extending outward indicate the region displayed by the sequence applet. Within the sequence display, the selected region is marked by a box around the feature label and highlighting of the nucleic acid sequence. Just below the nucleotides is a conceptual translation with three-letter amino acid codes, and a graphic indicating a stop codon. The GC content of the sequence is shown as a bar graph beneath the translation. Other features overlapping the selected region are shown as lines and labels. The text display on the *left* was produced by clicking the Selected Feature in Detail button in the map applet, and shows the text record for this feature.

output as well in a `/db_xref` qualifier, and the ATLAS Entry ID is shown in the entry-level comment. We hope to produce output in the more parseable ASN.1 or GIO formats in the future, so data may be exported more easily to tools capable of dealing with these common formats.

Query Interface

SEXTANT provides a number of predefined queries to retrieve individual or multiple Entry, Feature, or Experiment records. Entry-level queries allow selection by one of several IDs, origin, presence of particular feature types, arbitrary regular expression search, or location (limited to human chromosome 22). Feature queries may be constrained by Entry ID, Feature ID, Feature type, supporting evidence, or origin. Experiment queries are provided for Experiment ID, ID of a linked Feature, or Experiment type. All queries are specified via HTML forms. HTTP user authentication and cookies are used to grant access to confidential data.

Data Entry

SEXTANT also provides an HTML form for submission of new entries. In addition to the sequence itself, users may specify values for any other Entry-level fields. Fields that require a value are indicated on the form, and where values are constrained by controlled vocabularies, users are presented with lists. Access to the resulting ATLAS Entry, and all Features and Experiments based on it, may be restricted by the user. Basic consistency checks are performed when the form is processed, and errors are reported to the user. If the checks succeed, an ATLAS Entry is created, the CARTA scheduler is automatically invoked to run the default annotation

strategy, and the Entry ID is returned to the user. Framework annotation becomes available through the regular query interface as it is generated; at present it takes ~20 min to process a cosmid-sized DNA sequence.

Initial Experience

The GAIA prototype currently contains several hundred human genomic sequences retrieved from public databases, as well as a number of directly submitted sequences, primarily from human chromosome 22 (Table 1). The new entries consist almost entirely of finished sequence, both because the CARTA sensors provide better results over complete sequences and because the prototype database does not deal fully with discontinuous virtual sequences. Furthermore, we expect that once high-throughput sequencing is well under way, the time elapsed between production of level 2 sequence and finishing will be sufficiently brief that extensive efforts to coordinate annotation across multiple fragments will not be useful. It is interesting to note that records generated by CARTA account for the majority of Features on public database entries as well as new sequences. Approximately two thirds of these are similarity features; further work will be necessary to determine how often this represents discovery of a new transcript in known sequence, and how often it is an artifact or maps to a known transcript.

Our initial tests of automated annotation focused on a ~2 Mb-region from human chromosome 22 that was sequenced at the University of Oklahoma in collaboration with the chromosome 22 research group at the Children's Hospital of Philadelphia (CHOP). This region contains ~25 genes unevenly distributed across gene-rich and gene-poor regions. Aggregate statistics for annotation by

Table 1. Summary of Current Data in ATLAS

Origin	Entries	Nucleotides	All features	New features	Experiments
Retrieved	1152	16858822	75622	59895	11452
Submitted	97	5340872	16522	16522	1657
Total	1249	22199694	92144	76417	13109

Current totals are shown for several parameters describing the contents of ATLAS. The values for New features consider only those features added since an Entry was first deposited with GAIA; All features include those that were mapped into GAIA when a sequence was retrieved from GenBank. These values overestimate the true diversity of the database somewhat, as overlapping sequence from different entries or redundant features will contribute more than once to the totals.

BAILEY ET AL.

CARTA are shown in Table 2. When we compared framework annotation with the transcript map in a ~200-kb benchmark area (Gong et al. 1996) determined by extensive laboratory experimentation to contain eleven genes, we found that the automated annotation detected eight genes. Each of the three transcripts missed had only a single exon defined in the sequence, and lacked a clear ORF. Although these transcripts were supported by both the initial cloning experiments and subsequent Northern blots, it is unclear whether they represent parts of larger protein-coding genes or sterile transcripts produced in a region of active transcription. In some cases, only a part of a gene was detected by framework annotation, but this is consistent with its purpose: to identify regions of interest for further laboratory investigation, on which complete determination of gene structure can be based.

GRAIL was the single most sensitive sensor, detecting all eight transcripts (one on the strand opposite the actual gene), but also the least specific, with >70 predicted exons not overlapping a known transcript. *interprEST* yielded the best balance of sensitivity and specificity, detecting seven genes, but only three spans not overlapping a gene, two of which coincide with transcripts reported by another group but not detected by the experiments performed at CHOP (IC accession nos. X91348 and L42354). *KnSA* detected these transcripts as well as four of the CHOP transcripts and an STS, and often provided extensive information on gene structure through multiexon alignments, as well as clues about the function of the predicted gene. As expected, combination of results from multiple sensors frequently led to a more complete model than data from a single sensor.

Data from this region also demonstrate clearly the benefits of refreshing annotation over time. When the pattern of alignments produced by *interprEST* and *KnSA* is projected back over time, it shows that not only the number of genes detected,

but also the number of alignments per gene, and therefore the potential depth of information about the gene, increases with time (Fig. 4A). Comparison of GRAIL results at two points in time (Fig. 4B) points out the differences that revision of analytical engines may produce, including addition, disappearance, or consolidation of features.

Finally, framework annotation has proved its value prospectively, in that it has provided significant assistance in characterization of new genes. An example of this is drawn from a sequence in the mouse MHC region that was associated with a hormone-induced cleft palate phenotype. As reported previously (Bedian et al. 1997), framework annotation demonstrated a cluster of ESTs not overlapping a known transcript. This cluster provided an initial model of exon structure, including potential alternative splicing, which was extended by further sequencing of cDNA clones associated with the ESTs. Amino acid similarity search produced a strong alignment from which a functional prediction could be made.

DISCUSSION

Framework annotation differs from traditional sequence annotation in several ways: the need for high-throughput methods, a basis more in computational prediction than in description of laboratory results, the goal of identification rather than complete characterization of genes, and an emphasis on consistency and reliability of annotation across large amounts of sequence. These characteristics arise from the need to develop an initial structure within which to begin more detailed examination of new sequence produced by genome-scale initiatives. The GAIA prototype implements several principles of framework annotation, with a focus on human sequence, to provide a testbed for refinement of high-throughput automated annotation. The re-

Table 2. Summary of CARTA Processing for ~2 Mb of Human Test Sequence

Value	IRE	<i>interprEST</i>	<i>KnSA</i>	GRAIL	STS	Total
No. of features	1462	662	124	1196	75	3538
% CPU time	37.4	20.3	33.8	0.9	7.6	100

The number of features generated by each sensor is shown, as is the percent consumed by each sensor of the total CPU time expended on annotation. The absolute CPU time required by the computing resources used for this test was 19 hr, 7 min, 22 sec. Because of dependencies between sensors, the minimum elapsed time necessary to annotate the region was 13 hr, 37 min, 24 sec.

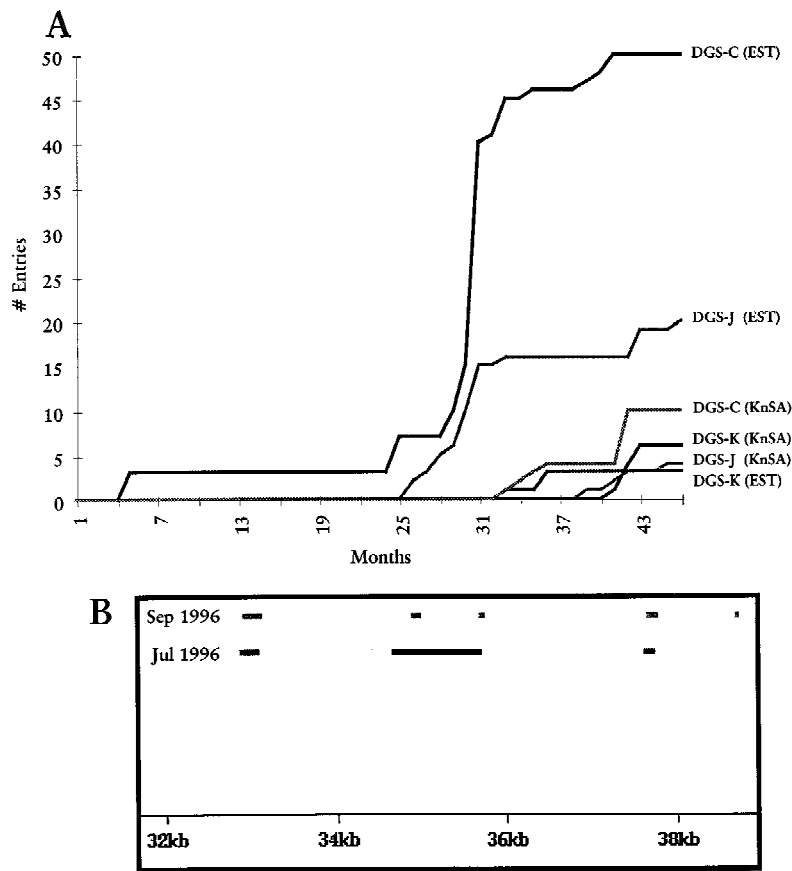


Figure 4 Change in annotation over time. (A) Accumulation over ~4 years of alignments from CARTA's similarity sensors with three benchmark genes in the DiGeorge critical region on human chromosome 22. The y-axis value indicates the number of target database (dbEST or GenBank) entries at a given point in time that met the sensor's threshold for feature creation. (B) Detail of GRAIL2 annotation in a portion of the DiGeorge critical region at two points in time, showing changes in GRAIL exon predictions.

sults presented here demonstrate the utility of this process in a set of test sequences.

A great deal of work remains to be done, however. As discussed above, the current ATLAS schema must be revised to describe more fully specific types of information, and more capable constraint-checking mechanisms are needed. It will also be desirable to move from the traditional entry-based data model to a view based on the underlying structure of the genome. However, this process will require handling of virtual sequences that may be incomplete or may specify more than one sequence for a given location.

The sensors present in the initial version of CARTA represent several major categories of analyses, but are by no means comprehensive, nor even

necessarily the best examples of their category. Further study of these methods on larger sets of sequences will help to better determine how well they generalize and how best to manage annotation over time. New approaches must be explored as well. In particular, peptide-based similarity search has been effective in identifying sequences conserved across large evolutionary distances, in many cases providing information useful for predicting the function of a newly discovered gene. Finally, it will be necessary to automate the integration of information from multiple sources when constructing gene models.

Although the current user interface begins to address issues raised by the need to display large amounts of annotation from multiple sources, a more flexible environment must be developed in which it is easier to pursue additional analyses and examine different regions or large sets of sequences. In the longer term, provisions must be made for information to be fed back from the community into annotation databases, while preserving the quality and consistency of these resources.

Most important, however, is the need for ongoing discussion in the genomics community about the role of systematic annotation. There has been a growing consensus that some systematic approach to large-scale annotation would add significant value to genome-scale sequencing efforts. This is a new endeavor, however, particularly with respect to complex mammalian genomes. Although a number of ideas have been recently put forth, experience with the process is still limited. We feel that a well-reasoned approach to the problem will be of greater benefit in the long term than a more rapid, but ad hoc, application of available methods under different circumstances.

The GAIA prototype is available for users to browse the current contents of the database and submit new sequences for annotation. We hope to expand its coverage and add interactive capabilities, to make it more useful both as a repository of developing knowledge and as a workbench for studying sequences of interest.

In the long term, though, annotation must be a

BAILEY ET AL.

collaborative effort across the community. More effective approaches to producing and accessing data must still be developed, and it is our hope that GAIA can contribute to the creation of effective community-wide tools for managing genome-scale sequence data.

METHODS

Computational Resources

Computation was performed on a collection of Sun SPARCstations and UltraSPARCs running the Solaris 2.4 or 2.5 operating system. Most of ATLAS and CARTA were written in Perl; C and SQL were used where appropriate. SEXTANT was written in Perl and Java. Publicly available software tools were compiled locally where possible; otherwise executable images were obtained from the authors. Public sequence databases were available via local flatfile copies, which were updated from servers at the U.S. National Center for Biotechnology Information (NCBI) daily, and periodically regenerated completely to insure synchronization with the master versions.

Annotation Database

The annotation database was maintained as a Sybase relational database. SDL expressions were stored as text fields, with some values copied into separate columns in the same table to permit faster queries. All queries were performed via the ATLAS interface routines, in which SDL queries were mapped into SQL, and access restrictions were applied. SDL results were parsed into Perl data structures, which were returned to the calling routine. At some points in SEXTANT, SQL queries were passed directly through ATLAS, and relational tuples returned, to permit faster response to queries if the data requested involved only fields broken out as separate columns in the relational database.

Sequence Entry

For sequences retrieved from GenBank, feature tables were parsed by SSP1, and the resulting files were used to construct ATLAS entries without further curation. Map location was taken from feature qualifiers or entry-level descriptions. New sequences submitted by the investigators were deposited either through the batch entry procedure or SEXTANT; all outside entries were created via SEXTANT. Gene-related features in new entries were added manually

on the basis of information provided by collaborators; these were linked to `author Experiments`.

IREsense

Interspersed and simple sequence repeats were identified with a binary image of CENSOR (Jurka et al. 1996) version 1.1 kindly made available by the authors. Repeat templates consisted of a collection of species-specific repeat elements (v5.0, Jurka et al. 1992), simple repeat elements (v3.0, Jurka and Pethiyagoda 1995), and the cloning vector pUC19. CENSOR was run once with a linear gap penalty and again with a logarithmic gap penalty, to better identify partial repeats; the default alignment parameters were used at each step. Alu repeats were subtended by invoking the `alu` program from the PYTHIA suite (Milosavljevic and Jurka 1993).

interprESTsense

Masked sequences were used as a query in a similarity search against all of dbEST by use of `blastn v1.4.8` (Altschul et al. 1990). Particulars of the search and postprocessing were as described previously (Bailey et al. 1998). Similarity features were created for alignments falling into stringency class 90L.

KnSA

Masking of repeats and BLAST searches were performed as described for `interprESTsense`. The species of origin for a sequence was determined by pattern matching on the `Source` field of the `Entry`, and the matching phylogenetic GenBank division was searched, as were the `patent`, `unannotated`, and `STS` divisions, and the incremental updates to `nontag` sequences. Alignments that covered at least 100 nucleotides of the target sequence, or its entire length if <100, at 90% identity were used to create similarity features.

GRAILsense

The species of origin for a sequence was determined by pattern matching on the `Source` field of the `Entry`, and appropriate `grail1a` and `grail2` queries were constructed and dispatched to the GRAIL server at the Oak Ridge National Laboratory (Oak Ridge, TN). Results were parsed, and predicted exons having a quality of `good` or `excellent` were used to create features.

STSsense

PCR primer sequences and product sizes for STSs were compiled from GDB and from the Human Chromosome 22 Genome Center map database. Primer sequences were aligned with the query sequence by use of the IUB alphabet to establish compatibility. An STS Feature was constructed if the primers aligned with opposite strands of the query, with no more than two mismatches each, and were separated by a distance within 20 nucleotides of the predicted product length.

Comparison of Framework Annotation with Experimental Transcript Map

Features generated by CARTA sensors were compared with manually entered gene-related features. Two features were considered to overlap if any positions on the sequence were common to both. Strand was not significant in this comparison.

Reconstruction of Time Course for Similarity Results

Identifiers for target sequences were taken from `similarity` features overlapping a given gene. The creation dates in the corresponding dbEST or GenBank records were used to determine when each sequence became available, and an alignment with a database entry was considered to be a positive result for the appropriate sensor from that time forward.

Changes in GRAIL Annotation

Identical GRAILsense runs were performed for the query sequence on the dates indicated, and selected exon predictions were displayed in a SEXTANT map applet, from which the figure was constructed.

ACKNOWLEDGMENTS

We thank Juergen Haas for assistance with the SSP1 parser, Shan Dong for initial implementation of the SEXTANT Java applets, and Marcia Budarf, Weilong Gong, Bruce Roe, and their laboratories for providing access to data from the DGCR prior to publication. We are also grateful to David Searls for helpful discussions, especially related to data visualization, early in the project. This work was supported by U.S. Department of Energy grant DEFG0292ER61371 (G.C.O.), and National Institutes of Health grants 1R01HG0153901 and R01HG0145001 (L.C.B. and G.C.O.). L.C.B. was supported by a fellowship from the Life and Health Insurance Medical Research Fund.

The publication costs of this article were defrayed in part by payment of page charges. This article must therefore be

hereby marked "advertisement" in accordance with 18 USC section 1734 solely to indicate this fact.

REFERENCES

- Altschul, S.F., W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. 1990. Basic local alignment search tool. *J. Mol. Biol.* 215: 403–410.
- Bailey, L.C., Jr., D.B. Searls, and G.C. Overton. 1998. EST-driven gene annotation in human genomic sequence. *Genome Res.* (in press).
- Bedian, V., T. Adams, E.A. Geiger, L.C. Bailey, and D.L. Gasser. 1997. A gene belonging to the Sm family of snRNP core proteins maps within the mouse MHC. *Immunogenetics* 46: 427–430.
- Buneman, P., S.B. Davidson, K. Hart, G.C. Overton, and L. Wong. 1994. A data transformation system for biological data sources. In *21st international conference on very large data bases*, pp. 158–169. Morgan-Kaufmann Publishers, San Mateo, CA.
- Burge, C. and S. Karlin. 1997. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268: 78–94.
- Burset, M. and R. Guigo. 1996. Evaluation of gene structure prediction programs. *Genomics* 34: 353–367.
- Charlebois, R.L., T. Gaasterland, M.A. Ragan, W.F. Doolittle, and C.W. Sensen. 1996. The *Sulfolobus solfataricus* P2 genome project. *FEBS Lett.* 389: 88–91.
- Crabtree, J., S. Fischer, M. Gibson, and G.C. Overton. 1998. bioWidgets: Reusable visualization components for bioinformatics. In *Molecular biology database* (ed. S. Letovsky). Kluwer Academic Publishers, Dordrecht, The Netherlands.
- DDBJ/EMBL/GenBank. 1996. DDBJ/EMBL/GenBank Feature table: Definition.
- GCG (Genetics Computer Group). 1996. Program manual for the Wisconsin sequence analysis package, version 9.0. Genetics Computer Group, Madison, WI.
- Gong, W., B.S. Emanuel, J. Collins, D.H. Kim, Z. Wang, F. Chen, G. Zhang, B. Roe, and M.L. Budarf. 1996. A transcription map of the DiGeorge and velo-cardio-facial syndrome minimal critical region on 22q11. *Hum. Mol. Genet.* 5: 789–800.
- Harger, C., M. Skupski, E. Allen, C. Clark, D. Crowley, E. Dickinson, D. Easley, A. Espinosa-Lujan, A. Farmer, C. Fields et al. 1997. The Genome Sequence DataBase version 1.0 (GSDB): From low pass sequences to complete genomes. *Nucleic Acids Res.* 25: 18–23.
- Harris, N.L. 1997. Genotator: A workbench for sequence annotation. *Genome Res.* 7: 754–762.

BAILEY ET AL.

Helt, G. 1996. Drosophila Genome Browser Prototype. <http://fruitfly.berkeley.edu/javademos/GenomeBrowser.html>

Jurka, J. and C. Pethiyagoda. 1995. Simple repetitive DNA sequences from primates: compilation and analysis. *J. Mol. Evol.* 40: 120–126.

Jurka, J., J. Walichewicz, and A. Milosavljevic. 1992. Prototypic sequences for human repetitive DNA. *J. Mol. Evol.* 35: 286–291.

Jurka, J., P. Klonowski, V. Dagman, and P. Pelton. 1996. CENSOR—A program for identification and elimination of repetitive elements from DNA sequences. *Comput. Chem.* 20: 119–121.

Makalowski, W., J. Zhang, and M.S. Boguski. 1996. Comparative analysis of 1196 orthologous mouse and human full-length mRNA and protein sequences. *Genome Res.* 6: 846–857.

Milosavljevic, A. and J. Jurka. 1993. Discovering simple DNA sequences by the algorithmic significance method. *Comput. Appl. Biosci.* 9: 407–411.

NCGR (National Center for Genome Resources). 1996. Genome Sequence DataBase Schema (GSDB) Version 1.0. Santa Fe, NM.

Ouzounis, C., P. Bork, G. Casari, and C. Sander. 1995. New protein functions in yeast chromosome VIII. *Protein Sci.* 4: 2424–2428.

Overton, G.C., J. Aaronson, J. Haas, and J. Adams. 1994. QCB: A system for querying sequence database fields and features. *J. Comp. Biol.* 1: 3–13.

Pangea Systems, Inc. 1997. GeneWorld 2.5. http://www.panbio.com/3_product/geneworld.htm

Scharf, M., R. Schneider, G. Casari, P. Bork, A. Valencia, C. Ouzounis, and C. Sander. 1994. GeneQuiz: A workbench for sequence analysis. *Intell. Systems Mol. Biol.* 2: 348–353.

Schug, J. and G.C. Overton. 1997. TESS: Transcription Element Search Software on the WWW. Technical Report CBIL-TR-1997-1001. Computational Biology and Informatics Laboratory, School of Medicine, University of Pennsylvania, Philadelphia, PA.

Searls, D.B. 1995. bioTk: Componentry for genome informatics graphical user interfaces. *Gene* 163: GC1–16.

Smith, S.W., R. Overbeek, C.R. Woese, W. Gilbert, and P.M. Gillevet. 1994. The genetic data environment an expandable GUI for multiple sequence analysis. *Comput. Appl. Biosci.* 10: 671–675.

Tomb, J.F., O. White, A.R. Kerlavage, R.A. Clayton, G.G. Sutton, R.D. Fleischmann, K.A. Ketchum, H.P. Klenk, S. Gill, B.A. Dougherty et al. 1997. The complete genome sequence of the gastric pathogen *Helicobacter pylori*. *Nature* 388: 539–547.

Uberbacher, E. et al. 1997. The genome channel. <http://compbio.ornl.gov/CoLab/channel/index.html>

Uberbacher, E.C., Y. Xu, and R.J. Mural. 1996. Discovering and understanding genes in human DNA sequence using GRAIL. *Meth. Enzymol.* 266: 259–281.

Wolfsberg, T., B.F. Ouellette, and J. Kans. 1997. Sequin. <http://www.ncbi.nlm.nih.gov/Sequin/>

Zhang, J. and T.L. Madden. 1997. PowerBLAST: A new network BLAST application for interactive or automated sequence analysis and annotation. *Genome Res.* 7: 649–656.

Received December 5, 1997; accepted in revised form February 5, 1998.