Philosophy and Definition for a Universal Genetic Sequence Database

Thomas D. Schneider *

version = 1.80 of philgen.tex 1994 June 29

ABSTRACT

Modern sequence databases have many problems because they are not carefully defined. For example, when one searches for homology with a given sequence, many duplicate sequences are found in each database and similar but not identical results are obtained from other databases. The extra copies of inconsistent information are slowing down research. This situation could easily be avoided by removing redundancy in the databases, but this goal is not a fundamental component of the database design and has been neglected. A clear statement of goals for storing genetic sequence information is required. To this end, five documents, in decreasing order of importance, are proposed:
(1) The Philosophy document defines guiding principles for the design and use of the database.
(2) The Definition document identifies what is to be stored in the database, following the guidance of the philosophy. It is machine and computer-language independent.
(3) The Implementation document translates the definition into computer code which can be run on machines.
(4) Examples of all database objects allow users to test their database analysis programs.
(5) A Tutorial explains how the database is organized.
A philosophy and the beginnings of a definition are given in this paper.

Individual researchers can help by:
1. providing correct and complete sequence data;
2. annotating their sequences with experimentally deduced features;
3. providing standard genetic names for all features;
4. updating the annotation as new knowledge is obtained;
5. periodically verifying the accuracy of the data and annotation in the public database;
6. volunteering to curate data for which they are an expert.

*National Cancer Institute, Frederick Cancer Research and Development Center, Laboratory of Mathematical Biology, P. O. Box B, Frederick, MD 21702-1201. Internet address: toms@ncifcrf.gov.

> GenBank runs the risk of becoming so much noise. It isn't just a matter of collecting all sequences like some avid bowerbird, researchers should be able to get information OUT of GenBank as well.—Ingrid Jakobsen [1]

> Unless the architecture and the nature of the information represented in a database are concisely and comprehensively defined, the database itself will be of limited value.—David George [2]

> At the time of writing, data-base technology is widely misunderstood. Its role as the foundation stone of future data processing is often not appreciated. The techniques used in many organizations contain the seeds of immense future difficulties. Data independence is often thrown to the winds. Data organizations in use prevent the data being employed as it should be.—James Martin [3]

INTRODUCTION

The way data are stored in a database affects the kind and quality of the science that can be done with the database. My scientific work has led me to a particular view of how genetic sequence databases can be organized to make them useful for research. Fifteen years ago I began work on *E. coli* ribosome binding sites by typing into a computer the sequences around the initiation codons of 63 genes. I soon realized that if I wanted to look at different regions relative to the starts, I would have to edit all the sequences by hand. This was likely to lead to many errors [4], and I would have to repeat the editing for each new analysis. To avoid that, I decided to store all the known sequences and then to write a program which could extract the portions I needed for a particular analysis. To tell the program what I wanted, I invented a language for doing the extractions and named it Delila, for DEoxyribonucleic acid LIbrary LAnguage [5, 6, 7]. In this language one first specifies a sequence region:

```
organism S.cerevisiae;
chromosome III;
gene LEU2;
```

Then one requests particular parts of that sequence:

```
get from gene begin -30 to gene begin +40;
```

This will extract the sequence from 30 bases before the first base of the initiation codon to 40 bases after it. This instruction is "relative" because it defines a sequence relative to a defined chromosomal location. Alternatively, one can give the absolute coordinates:

```
get from 90935 -30 to 90935 +40;
```

or equivalently

```
get from 90905 to 90975;
```

Editing instructions is much easier, safer, faster and more reliable than editing sequences directly. In addition, other programs can create or modify a list of instructions, so powerful ways of manipulating large sets of sequences became possible [8]. The Delila program has been enormously useful for many studies. In particular, it formed the basis for the first use of neural nets [9] and information theory [10, 11, 12] to analyze binding sites. In a recent analysis we manipulated almost 3600 sequences by this method [13], and in unpublished work we have easily worked with 8000 sequence fragments.

At about the same time that Delila was being created, the international sequence databases were formed to reduce the worldwide duplication of effort in capturing sequence data. The databases have been highly successful in this endeavor. It would be very useful to write a "Delila II" program which could take full advantage of the larger databases. As with Delila I, Delila II could use the names of genetic features to define the sequences of interest. Since the location of many genetic features is well defined, or could be defined by international convention [14], this approach would make the Delila instructions reasonably insensitive to changes in the growing database. Scientists around the world could exchange their lists of sequence instructions, and could build on each other's work.

As described in the next section, the present structure of the GenBank database [15] makes these kinds of powerful sequence manipulations difficult. We are gathering sequences at an exponentially increasing rate, but the locations of genetic objects are not being stored so that they can be retrieved automatically. Because there is no overall idea of how the data could be used, the data are not being stored in a sensible way. Just as we once had a crisis about capturing sequence data, we now are approaching a crisis about compiling and annotating sequences.

There are two basic reasons for these problems. First, there are no published principles guiding the design of the database. Second, there is no published and accepted documentation that defines the entire database. Without a written document, a researcher cannot recognize that the database structure is incorrect. Documentation that does exist is often incomplete and unclear. The documentation usually describes implementation of the database and does not say why the database was built this way. There is no defined set of rules by which the database is run. The content and quality of data varies.

To solve these problems, this paper proposes the adoption of a collection of guiding principles for sequence databases. A system of fundamental or motivating principles is a "philosophy" [16]. Without a guiding philosophy we will eventually be unable to use much of the data we are spending millions of dollars to collect [17, 18].

A philosophy is not enough. We must use the philosophy to precisely define the contents of the database. We must publish the definition so that the ideas can be widely disseminated and critiqued. The importance of documenting computer programs is widely understood; the same should apply to databases. Examples of such documents are given in the references [5, 19, 20, 21].

I do not merely mean that we need manuals which explain the database to outsiders or describe its current status. Instead, I envision documents equivalent to the constitution and laws of a country. They guide the course of events. They are "alive" and continuously debated. They are changed when times change. But if they are not changed, then they are strictly followed. GenBank needs such guidance.

This paper also proposes that we should store genetic sequence information in a way close to that found in nature. This would allow the sequence information to be gathered into huge but well structured and easily understood data objects. Instructions for obtaining those objects would be easy to write, and instruction lists could be exchanged between working scientists. Attaining a clean and organized sequence database will require the concerted efforts of not only all the database staffs to precisely define the data structure, but also those of every molecular biologist who works with sequence data to provide and check the data in the structure. Design and development of advanced database access languages such as Delila II will be blocked until the database is clean.

DATABASE PROBLEMS: a "Taxonomy" of Errors

There are many kinds of errors in the GenBank database [22, 23, 24, 25, 26, 27]. We discuss some of them here, along with examples, to give the reader an idea of the enormous magnitude of the problem and to explain how these problems affect designing and using Delila-like sequence access languages for scientific research.

- **Sequencing errors** [28]. Surprisingly, coding sequences with stop codons in the middle are frequently sent to GenBank, as are DNA sequences whose translations do not match the translation submitted by the author (D. Lipman, personal communication). Although information theory and statistical methods can be reasonably insensitive to errors, they can cause difficulties for training neural networks. Conversely, networks can be used to discover errors in the database [22], but obviously these must be corrected using experimental data.

- **Spelling errors and inconsistent names.** If objects in the database are not named consistently, then an instruction such as: "get all Homo sapiens liver expressed donor splice sites from 50 bases prior to the junction to 50 bases after the junction" will not give all the available data.

- **Missing data.** During our analysis of splice junctions [13], we were shocked to discover that the amount of sequence provided by a researcher $5'$ to each acceptor site is often a multiple of 5 bases (Fig. 1). This could not be a natural occurrence because ⇐Fig 1 the ends of sequencing runs are not perfectly correlated with the ends of introns. Evidently researchers are not reporting all of their data to the database. This weakened our statistical results. Further, since GenBank concentrates on new sequence data, feature information found after publication of a sequence rarely goes into the database. Much of our detailed biological knowledge is being buried in the paper literature and is unavailable for automatic retrieval.

- **Parsing problems.** Data has to be in a regular format for programs to use it. As a GenBank advisor, about 10 years ago, I pointed out that the feature data could not be split into its smallest components by a computer program, and that a human was required to extract data from the database. No "artificial intelligence" is intelligent enough to do the extraction. Amazingly, entries were not automatically separable into parts (parsable) even last year. However, a specialized data language called Abstract Syntax Notation One (ASN.1, [29]), was recently introduced to finally resolve this issue. Thus there appears to be a modern solution. Yet ASN.1 is not enough, as it does not begin to resolve the remaining fundamental issues. For example, it is still impossible to automatically extract the mutations of bacteriophage lambda because the data on mutations are stored in unparsable "notes" having inconsistent formats. These cannot be automatically captured by programs to generate the mutant sequences. The types of analyses that may be performed are restricted. One cannot, for example, automatically construct a series of mutations for analysis of the effect on the folding of mRNA. This impenetrable storage method won't be changed by translation to ASN.1.

- **Sequence labels are arbitrary and unstable.** Unfortunately the sequences in GenBank are not stored by their genetic location. Instead, the data is stored as overlapping and redundant reports of sequences called 'entries'. Although we would prefer to use genetic names to identify our sequences, we are forced by the database structure to use the entry (LOCUS) names. Unfortunately this is only the name of a sequence, not a true genetic locus. When LOCUS names change, our Delila instruction sets become outdated. This database organization forces us to write Delila instructions which become useless within a year. The instructions cannot be used for reliable communication between laboratories. ACCESSION numbers have similar problems [29]. Even the most recently invented GenInfo (GI) numbers do not solve this problem. GI numbers are supposed to refer to the same sequence forever [29]. When a sequence is changed, a copy with a new GI number is created. The new sequence has a pointer to the old and *vice versa*. This means that the sequences obtained from a list of GIs will not get automatic corrections, thus assuring stability of the data being accessed. Yet that also means that my GI list of splice junctions would slowly become *scientifically* obsolete as sequence corrections are made. The GI approach also does not allow our list of splice junctions to be automatically improved as researchers deposit more data (for example, to fill in Fig. 1). Instead, special software would be required to keep updating one's instruction list. It would be simpler, more stable, and far more natural to list the biological names of the objects relevant to an ongoing study. For *each particular analysis* I could automatically obtain the GI list as a record of the exact data set used. This could be implemented on top of the GI scheme by having the named data objects always refer to the most recent GI number. In addition, numerical tags are much more difficult to remember than names.

- **Duplicated entries.** If there are two copies of a piece of data in a database, then there is a good chance that one of them may be updated or corrected while the other is not. This leads to inconsistency in the database [3]. For example:

  - There are 4 complete copies of the first widely used cloning vector pBR322 [30], in the current GenBank database (L08654, J01749 (two different dates), and V01119 in 82.0, 4/8/94) that differ in length. (In 1993 there were only two copies.)
  - There are 2 complete copies of bacteriophage lambda (V00636 and J02459) which differ by 4 mismatches. There are many other overlapping entries.
  - There are 3 copies of the transposable element Tn5 sequence, U00004, L19385 and L19386. The word "Tn5" appeared in 57 entries of GenBank 75.0 (2/10/93), but the complete sequence *which has been available for years* was not included in this set. After merging these sequences, Paul Hengen discovered by chance that the complete sequence had been assembled by one person and sent to others by electronic mail, *but it was not donated to GenBank*. If the effort of this person or the originating laboratories had been directed properly, a single clean data object would have existed in the database, and other people's efforts would have been saved. Dr. Hengen and Andrew R. Bradbury completed the entry by merging all of the other entries, but the other entries still exist and get in the way of research.

– In GenBank 76 V01524 appeared in two entries; M18262 appeared in two different entries that had the same LOCUS name; X52972 appeared in two different entries with different LOCUS names, one of which is one of the two M18262 entries. Every person who looked at this confusing set of sequences was forced to figure out their relationships all over again since the ACCESSION number does not give this information.

There are an estimated 1000 or more such cases that the National Center for Biotechnology Information (NCBI, Bethesda, MD) has identified for clean up, now that NCBI is responsible for GenBank (J. Ostell, personal communication). The most common consequence for biologists is multiple "hits" obtained during sequence searches. These repeatedly waste a researcher's time and prevent one from easily seeing some important matches because searches always limit the degree of matching. Duplication also blocks surveys of the database because each researcher must comb through perhaps thousands of duplicated and overlapping sequences to obtain a clean set for analysis. Because of name changes, this tedious process must be repeated in its entirety every time a new version of the database appears. Automated cleaning methods may make errors or will miss useful data.

The GenBank database is kept in two parts, the old entries and the new and changed entries. The changes are periodically incorporated into the database. When one asks for sequences by electronic mail, one receives both the old and the new entries. From the user's viewpoint, all recently corrected sequences are duplicated!

When duplication is found, the researcher then must look up the original papers and try to decide which is correct. Typically the researcher will not tell the database staff the results, so every other researcher has to repeat the effort. Clearly all this clerical work cannot be performed if a single researcher wants to manipulate 100,000 sequence fragments!

- **Unmerged sequences.** These are sequences that overlap, and have not been joined into a single sequence. Doing this requires an expert, but it is absolutely necessary to allow the development of Delila-like access languages. How can we handle differences in the sequences? This is easy: create a feature which indicates what the difference is in the two reference sources. If the feature is recorded properly, the access program will be able to reconstruct either of the original sequences. How can we handle polymorphisms and strain differences? Record them as features with the name of the strain in each case. This would allow the one to specify the desired strain using the access language.

  A severe example of unmerged sequences is the 315357 bp yeast chromosome III [31] (Accession X59720 in GenBank 82.0). 95 entries other than the complete sequence have a source organism *Saccharomyces cerevisiae* and have "III" somewhere else in the entry. Because chromosome location is not consistently recorded, not all of these are on chromosome III. (For example, this set includes subtilisin-like protease III.) There is no way at present to automatically locate just the entries relevant to chromosome III. Conversely, features, such as threonine tRNA, are not recorded in the larger entry. As a result these known annotations are often missed. The authors could have merged all these sequences together to create a single entry.

- **Incorrect features and undocumented features.** Computer searches are based on a model of nature. Computer search results should not be stored in the database without a notation of the program and version number used to find the feature. Likewise, if we cannot obtain references to the experimental data used to identify a feature, we should not use it in statistical analyses. This places a huge burden on those who want to analyze various features because they have to remove all the "guesses" and putative features from the experimentally verified ones.

- **Incorrect feature placement.** In 1993, several exons had lengths of zero or even -1. (Because they "went under" and now require a certain amount of cleaning up, we dubbed these "Valdez Exons".) For example, the mRNA sequences in ACCESSION numbers M77774 through M77778 and J05046 have exons marked off. I detected these cases because the ends of the exons had "abnormal" splice junctions: they had already been spliced. In this case the junctions should have been noted, but not by the use of the type 'exon' because that is inconsistent with the usual use of 'exon'. (There are other fundamental problems with this entry: the authors knew the junctions, so they apparently failed to report known sequence data of the introns. Also, mRNA is derived from DNA and so should not be in the database in the first place.) This case demonstrates that subtle definition problems must be carefully addressed so that large scale surveys are not fouled. Since the database must be searched by computer programs, it is extremely important for the types to be consistent throughout the database.

  One consequence of incorrect feature placement is that it prevents one from writing general instructions such as "get all Saccharomyces cerevisiae exons from beginning to ending".

- **Useless features.** Data stored in a "note" is unavailable to Delila-like programs because the information cannot be extracted. Likewise, "misc" features are so vague that they cannot be used for writing meaningful instructions. My favorite case in the mystery department was in LOCUS RATTNT, ACCESSION M15202:

  ```
  misc_feature    3616..3628
                  /number=3
                  /note="potential pseudo [4]; putative"
  ```

  The "misc_feature" of the current implementation is not acceptable by the criterion that it is not precise. Although it would be best to eliminate them, this would not allow new data types to be recorded. I suggest that notes and misc features be used only rarely. *They should also be temporary and eliminated as soon as possible.*

- **Unnamed features.** A feature without a name cannot be reliably located later on because when the sequence is altered the numerical coordinate is no longer the same. If every feature had a name that was consistent, by international convention, then it could always be located again in the future. Lists of old names or synonyms could easily be recorded. Features are rarely named in GenBank, but this is essential for access languages.

- **Large scale features constructed from absolute coordinates.** Coding sequences are currently constructed by defining a series of absolute coordinates instead of referring to the names of simpler features. The first consequence of this design is that the junction locations given in the coding sequence are redundant with the locations of introns and exons. This means that correction of one may often leave the other uncorrected and so the database has inconsistencies. A program cannot be used to correct this. The fundamental biological features are the donor and acceptor splice junctions. If these were named and recorded, the introns and exons could be defined simply by giving the correct names. The entire coding region could then be constructed by referring to the names of the introns and exons, and alternative splicings would be easy to see. Correction of the location of a donor site would immediately correct all of the larger structures.

- **Lack of map information.** Entries in the database do not consistently have genetic map locations, so one cannot ask to "get all Homo sapiens chromosome 17 sequences".

- **Redundancy and inconsistency between databases.** We were using LOCUS PMUGINMOM for an analysis and, since LOCUS names are unstable, we changed the identification to the corresponding ACCESSION V01463. To our astonishment, the entry changed completely. On inspection, we found that when we first obtained the entries a year earlier, there had been a different entry having the same ACCESSION V01463 (LOCUS XXMU01) at the same time as PMUGINMOM. In the current version of the database PMUGINMOM was deleted. This entry had diverged between the EMBL and LANL entries, and NCBI deleted the one from LANL since the entry originated from EMBL (D. Lipman, personal communication). Unfortunately the sequences, references and features were not identical and the region we are interested in was completely lost in the shuffle.

Readers are encouraged to report the errors they find in GenBank and to make sure that the correction appears in the database. The current electronic mail address for error reports is update@ncbi.nlm.nih.gov.

These are difficult problems and we cannot expect them to be solved overnight. One step forward is to clarify our goals so that we can work together toward solutions. The following Philosophy and Definition are proposed as a starting point.

A DATABASE PHILOSOPHY

Each principle strongly affects how one goes about designing and constructing a database, but the detailed design is not given by the principle. Principles are like axioms in geometry, while (ideally) the design and implementation are like theorems built up from the axioms.

Principle 1: There is only one official copy of each piece of information in the primary database.

This principle implies that all other unofficial copies must be derived automatically. If duplications were allowed, then one duplicate could be modified without update of another. This leads to inconsistencies. Inconsistencies lead directly to errors when the wrong item is deleted or to data loss when an item is deleted which happens to carry non-redundant data [3].

When two authors publish overlapping sequences, a single merged view of the data should be created which contains *all* features in one place. Discrepancies should be recorded so that each original publication can be reconstructed automatically. In the present system, every scientist must perform the merge, and must locate all the differences because there is no guarantee that a merged view exists. Worse, when a merged sequence *has* been generated, the original entries are still kept, so the poor researcher must fight even more duplication. The multiplication of this wasted effort is enormous on a world-wide scale. Conversely, funding for curators (people who merge, correct and annotate sequences) is highly economical because a single person can save effort on the part of many people around the world.

A common objection to this proposal is to ask whether one can trust the person who does the merge. This is invalid, not only because a person doing such a merge must be an expert in the relevant sequences, but also because attempting to merge sequence data almost invariably reveals inconsistencies in the sequence data itself. By resolving these conflicts, the data are improved. In the current plans of NCBI, the original fragments are kept and can be used to generate a merged sequence [29]. This allows any researcher to confirm the merge, but also allows one to see it as merged. However, Principle 1 implies that it would be better to have only one copy of the data, and to produce the original unmerged and erroneous reports upon demand. The database would become easier to use because there would be far fewer sequences and they would be more carefully reviewed. Furthermore, if sequences are kept separate and merges are to be performed automatically, the failure mode is unmerged sequences, which is not what most biologists want. Conversely, if sequences are kept merged, computation is only needed when the original sequence is requested. This looks exactly like a Delila extraction ("get sequence of reference 6"), and therefore would be easy to do.

Principle 2: A user should not be required to extract the original paper(s) in order to work with a sequence.

The user should be provided with complete information describing each database object so that they can access, manipulate, process, do statistics, read or check a sequence. When one is manipulating thousands of sequence fragments [13], it is impractical to look up every relevant paper. Even if one could obtain and read all the papers, the current doubling time of 1.8 years for the database makes it impossible to keep up to date.

This principle implies that the database should capture biological knowledge so robustly that users will never feel that they must go back to the literature to learn the features of a sequence. 100 years from now nobody will want to open thousands of dusty and cracked journal volumes or scan thousands of feet of microfilm to read the original papers—they will want to look directly in the database, and they will want to be able to automatically identify every type of sequence object.

Principle 3: The database structure should allow derived subsets of the database to have the same form as the original database.

This allows one to use the same tools on the subset as on the original set. Any analysis of sequence data can be thought of as using a subset of a large database, even if the subset contains only one sequence member. For example, a subset could be the sequence of the plasmid pBR322. With a simple program, a subset of *that* subset could be created which contains *Hae*III restriction digestion fragments. If this principle is followed, it would be easy to apply the same program again to the *Hae*III fragment subset but with *Hpa*II to produce a double digest. Another kind of useful subset is a set of aligned sequences. *Every kind of sequence analysis can be performed on subsets of the database* so tools for creating subsets should have highest priority, and the database should be arranged to facilitate extraction of subsets, as in the Delila system [6, 8, 9, 7, 13]. The current database was not designed to permit the construction of subsets. This makes it difficult to analyze small regions embedded in a large sequence such as a whole chromosome [31].

Principle 4: Scientists are primarily responsible for the quality of their contributions to the database. Database editors and reviewers arbitrate the final form of each data object.

Each person who submits a sequence is responsible for its completeness and accuracy. It is impossible for a database staff to ensure this. Furthermore, as the scientist learns more about a sequence, he or she is obliged to enter the new data. If this is not done, the scientist's work will be effectively unavailable to other scientists because data in the original papers cannot be found by electronic searches. As additional data are contributed by other investigators, an editorial/referee process similar to that of major scientific journals could insure that any discrepancies are resolved and that new data are properly incorporated or merged. The community must deal with issues such as how many errors in a sequence of a given length are acceptable, what mechanisms will be used to detect errors and misinterpretation of natural variants as sequencing errors.

Principle 5: The primary responsibility of the database staff is to provide the organizational structure of the database. That structure must include 5 kinds of written documentation: a philosophy, a definition, an implementation description, examples and a tutorial.

> "This [not only] includes the physical organization of the data themselves ... but also includes providing a well thought out plan concerning how contributing scientists will interact with the database and for developing procedures for ensuring that these efforts are effectively coordinated." —David George [2]

When database design is mixed with implementation and the philosophy is neglected it is difficult to know what the overall strategy is or how a database is likely to be represented in the future. A set of easily accessible documents which thoroughly address these issues would solve many problems. These documents should continue to be developed as advances in computer technology and molecular biology are made.

1. **Philosophy.** The philosophy of the database describes the principles which guide its organization. This lets everyone know WHY the database has been structured as it is. *The philosophy must be written out in detail because only then can people challenge the fundamental ideas implied by the current implementation.*

2. **Definition or Design.** The definition document describes WHAT kinds of data are in the database, and their organization. This includes a database *schema* which diagrams the interconnections between the data items [3, 5]. The principles that comprise the philosophy define a framework for designing the database but there can be many designs that match a given philosophy. The organization of a genetic database should allow input, modification and retrieval of information, yet it should also present the data in a form corresponding to the user's mental model of genetics. Without these characteristics it is difficult to write efficient programs which access and manipulate the data and it is difficult to answer biological questions. All data types should be defined in detail. The way that the data are to be stored physically is *not* part of the definition. It is, however, necessary for the author of the definition document to create definitions that can be implemented efficiently.

   This document has not existed for GenBank, and as a result there is no agreement as to what should be contained in the database and how the data should be organized. This led to inconsistently used feature types and varying degrees of annotation. The entire database must be defined, not just the "features" [21].

3. **Implementation.** HOW the database is constructed in a specific computer language is described by this document. It is possible to have several different implementations for the same philosophy and definition. 100 years from now the implementation may well be different, but the design could be the same.

   The following kinds of questions are answered by the implementation document: Should the database be relational (*i.e.* like a table of numbers) or object oriented (*i.e.* like a set of nested objects)? Should storage be in flat files (*i.e.* a linear set of characters) or as structures resident in memory (*i.e.* always in active computer memory)? Limitations on the lengths of names and other implementation-dependent decisions are described. Syntax may be defined in languages like BNF [32] or ASN.1 [29], but *these do not address the philosophy or definitions issues.*

A good implementation-free database definition allows easier migration to new computer platforms [6, 7, 20].

Maintaining a clear distinction between implementation, design and philosophy is critical for the long term usefulness of a database. Implementation of a database without a clearly stated definition and philosophy is a poor engineering practice that lead to the problems described in this paper. Just as one draws plans before building a bridge, one should write the philosophy and definition *before* implementing any computer program. If changes are required, the definition should be altered before the database or code is modified. This was done for the Delila system [5]. This disciplined approach has several healthy consequences:

(a) The database or code is *always* fully documented.

(b) Other people can comment on proposed changes before they are implemented.

(c) Changes are made only after careful consideration and users are well warned of impending changes.

What can be done given that we already have a database, but don't have these documents? The appropriate response is to step back, set aside all political [33], commercial [34] and other considerations and develop the philosophy and design from a fresh viewpoint. Current database implementation can then be guided toward the new design.

The philosophy, definition and implementation documents each serve a different purpose and should be distinct separate documents. Existing documents mix the highest definitions of database structure and philosophy with the niggling details of what character should go where in a particular data object [20, 29, 21]. Not only is this confusing to the reader, but it completely obscures the overall strategy. If the document is unclear, it is difficult to recognize the problems associated with it, and therefore difficult to suggest corrections. The absence of a full set of documentation is a disservice to the end user. Maintaining a clear distinction between implementation, design and philosophy is critical for the long term usefulness of a database.

4. **Examples.** Examples of every database object should be given so that scientists can see how to express themselves in the database language. A small test set containing every defined data item would be most useful for the development of applications that call upon GenBank data.

5. **Tutorial.** This document describes how to create and maintain the database so that anybody can retrieve information from it and help to build it.

Most of the problems with GenBank described earlier in this paper could have been prevented if the documents described above had been created and an extensive series of programs designed to check the integrity of the database had been written and used. Such check programs must be written based on a defining document, *not* from the current implementation of the database, because that is not definitive. A test of the defining and implementation documents is that they allow anybody to write a program to check the database structure.

Principle 6: Everyone using the database is responsible for reporting errors, however small.

If an error is not reported, it will, of course, remain in the database where it could lead to erroneous biological or medical interpretations. Suppose that a gene is sequenced from two individuals, one of whom is a normal homozygote and the other of whom is a carrier of a recessive allele. Both sequences are reported but the recessive variant is not documented as a mutation, although this was known from another source. Someone with the disorder is found to carry two copies of the recessive allele, but because of the incorrect annotation in the database, that individual is not recognized to be expressing a mutant gene product, but is simply thought to carry a natural polymorphism. This could have serious consequences for diagnosis and treatment of such a patient. It is conceivable that both the database staff and the authors could be held legally responsible. We need to find incentives that encourage investigators to report corrections.

Principle 7: For determining scientific precedence, release of sequence or object data into the database has higher priority than physical publication.

Placing data into the database is a form of electronic publishing [34] that establishes precedence.

> "That is if you were the first to clone and sequence a particular gene, the next person who sequences the homologue would find your sequence already in the database, and would be obliged to cite your entry, even if it had not yet been published."—Brian Fristensky [35]

This principle is actually a variation (or corollary) of Principle 1, in that it gives the highest authority to the unique copy of each data item in the database. Publication of the data in other media has lower authority.

Sequence objects that are not entered into the database should not be considered discoveries, just as an invention is not a patent until it is published by the Patent Office. For example, suppose that a DNA segment has been sequenced and inserted into the database. Two years later someone confirms a predicted protein sequence and publishes it but does not update the database. Others who are interested in studying thousands of genes simultaneously do not have the resources to search for this additional data, and therefore don't use it or are forced to redo the experiments. Credit would not be given to the original work. Unannotated information will be lost or inaccessible in the coming electronic age.

This principle does not deny that the original publication contains valuable information and provides insights into the structure and function of the sequence that is described in the database.

Principle 8: There is only one biology.

It is well recognized that the biology of other organisms is similar to human biology. The pervasive use of nucleic acids, the central dogma and general metabolism point to the unity of biology, while general physical principles and evolution will undoubtedly apply to biological systems on other planets [36, 37, 12]. Biology is universal.

This principle has two consequences.

1. **There should only be one international sequence database.** The plethora of formats makes sequence analysis much more difficult than it should be. Submissions to multiple databases increases the likelyhood of redundant and inconsistent data. Politically motivated squabbling over database format and over who should accept data submissions should no longer be tolerated by molecular biologists. To maintain consistency, there should be only one submission point and only one dispersion point, overseen by an internationally supported standards committee [14].

   This does not mean that all the work should be done in one place and that there should be only one group of people doing database work. To the contrary, there is so much work to be done that much of it should be spread out to other data centers. But this should be controlled from a single point to avoid duplication of effort [33]. Loose federations of databases, advocated by some [38], would cause intractable inconsistencies. '...the community would be better served by the convenience of "one-stop shopping" ' [14].

   All data which can be associated with a sequence should be directly accessible in a single unified format and within a single data structure. The profusion of specialized databases of *different* formats limits the usefulness of computer tools that are applicable to many biological problems [39]. Close linkage between nucleic acid, protein, structural and other databases is *only a first step* towards a complete and uniform synthesis of molecular biological data.

   This is a variation of Principle 1, but applied to whole databases. Derived databases should be *automatically* created from the common database so that they can be kept up to date with no human intervention (Principle 3). This would ensure that complete and consistent annotation is in the subsets and we will no longer be forced to navigate through a forest of inconsistent databases.

   The lack of an internationally accepted standard format means that scientists from different parts of the world have a difficult time exchanging data.

2. **The database can follow a natural taxonomic and genetic organization plan.** The current databases are oriented towards human efforts. Thus the concept of an "entry" in GenBank is a reflection of individual researchers' efforts at obtaining a sequence in a particular genetic region. As more sequences become available, they are all thrown

together in a messy pile of entries. Everybody who wants to do thorough work is forced by this disorganization to run programs to figure out where the overlaps are and is forced to read many papers to determine how or whether the sequences are related. They are also forced to merge the features themselves.

The current implementation of GenBank emphasizes sequences and neglects new information about the sequences—there is no reliable mechanism for capturing information not associated with the original sequence. Information from different scientists must be merged together, yet it must be possible to trace back to the original data in every case. However, *being able to trace back is much less important than providing a clean and merged view of the data for scientific research.* The justification for creating the sequence database is scientific, not historical, research.

In nature, sequences have natural packages. The primary one is the taxonomic classification focused on species. Thus we should look at a database and see a series of data objects representing species and particular strains. Each species has a set of chromosomes. As sequencing proceeds, the chromosome sequences are filled out, *but this view of the data does not change*. When the chromosome is finished, there would be a *single* data object in the database rather then a pile of disconnected data like yeast chromosome III. For example, this is the organization of the Online Mendelian Inheritance in Man catalogue [40].

One common objection to a natural scheme is that it is only one "view" on the data. However, the natural viewpoint (as described more fully in the next section), is the one that biologists take of biology, and fundamental biology does not change rapidly so this is the most stable possible representation of the data [3]. Representing the original publication—as is now practiced—is unstable because it ignores new data. There is no robust mechanism for capturing new feature data and corrections into the current GenBank. Following Principle 3, any desired subset (or "view") could be created from a natural scheme. For example, if we wish to study all of the cytochromes, we would simply list the species, strain, chromosome (if known) and genetic location of each one. For careful scientific research one would always want this list, so why not make it play an active part in obtaining the data? *Such a list of names can be made stable by an international standards committee, and it will remain stable as the data coalesces.* In contrast, the current schemes eventually destroy the usefulness of every name list created. Scientists are unable to exchange stable lists or to update each other's lists. This slows down research.

A frequently espoused "view" of the data is the sequencer's own scientific contribution. This could easily be created from the merged data by running a simple program. Although we may feel proprietary attachments to the data today, 100 years from now nobody will care who sequenced what. So we may as well begin now to store the data in a natural and objective form. The longer we wait, the more difficult this task will become.

One question that commonly arises is how we should handle polymorphisms, strain differences and mobile genetic elements. The simplest way to do this is to store a single canonical sequence and then to record all the changes necessary to create a particular strain automatically. This appears to raise difficult questions. Which of the possible sequences should it be? Who will decide this? Surprisingly, it doesn't matter. We can overlap all known sequences to generate a canonical collage. By recording the strain information carefully, a program can generate any particular pure strain we want.

A DATABASE DEFINITION

The preliminary definition given below is for the organization of natural sequences. A clean way to handle artificial constructions, which conforms to Principle 1, is to define a language in which one can specify the natural and synthetic sequences which are to be joined together [19]. It would then be possible to store only the instructions for creating the constructs. This would allow the artificial sequences to be created upon demand, so they would not require redundancy or much storage.

Following Principle 8, the remainder of the database should be organized into a nested series of named objects: species, strain, chromosome, genetic locus and individual sequence object or sub-locus. A "schema" is a model of the data one wishes to represent in a database [3]. Rather than developing a complete formal schema of a natural organization, Fig. 2 and Fig. 3 show an example of the general concept. ⇐Fig 2 ⇐Fig 3

Sequence Object Definition: A sequence object is a (possibly discontinuous) region of genetic material with distinct properties. According to Principle 2, the data defining an object and associated with that object must be complete. New information about the object must be continuously added to the database to keep it up to date, even if no new sequence information has been obtained.

1. **Every object has a type.** A rigidly controlled list of types must be defined. New types are added as new biological features are discovered. The types must form a logical, non-overlapping set of definitions of biological objects. Each type must be defined as part of the written definition and carefully distinguished from related types so that objects are not assigned the wrong type. Having a consistently defined type makes it possible to automatically obtain a comprehensive list of objects for study. Clear, written definitions are absolutely required. A complete list of types and their definitions is beyond the scope of this paper. For the current GenBank definitions see [21].

2. **Every object has a name.** For a computer to find and manipulate an object, the object must have a name which is unique within a certain scope. Complete names have 4 parts consisting of the types: *species, strain, gene locus* and *specific sequence object*. For example *E. coli, K12, lac, Z* refers to the *lacZ* coding region. (Chromosome names are optional because genetic loci are, by convention, unique to whole genomes.) The species and strain information must be associated with the object, since we must be able to perform manipulations which create chimeric sequences, and the origin of those sequences must be maintained. Names allow one to define genetic locations RELATIVE to an object. This has the extreme advantage that the position of the object may shift, but one's instructions for obtaining the object would not be affected. For example, after specifying the object mentioned above, we can then say that we are interested in the region from -60 to +40 around the beginning of *Z* (*i.e.* the ribosome binding site). This particular instruction has specified the same sequence for many years, and this will remain true even as the entire sequence of *E. coli* is being completed. In contrast, an

absolute position (*e.g.* 314159) written down by a user of a database subset is useless as soon as sequences are merged because at least part of the sequence must be renumbered.

It is possible to assign each data object an official multi-part name. The most logical and useful starting point for choosing a name for an object is its standard genetic name [14]. Although these names may change as standards become better, they are more stable than anything else. *The database defines the international standard names.* A list of synonyms for outdated names could be associated with each object. By this means, one may take an old list of objects and use it for the most part in a later version of the database. This would allow scientists to exchange reliable, stable lists that define subsets of the database, and therefore it is extremely important for the future of sequence analysis.

In computer programming languages, such as Pascal [41], names only apply to a certain region of code called the scope of the name. The same concept already applies to genetic names (Fig. 2, Fig. 3). The widely accepted convention is that all genetic locus names are consistent within one species so that each name only refers to one locus. Because the scope is limited to a species, there is no confusion as long as the species is always given as part of the name. Names within a given genetic locus should also have their scope limited to that locus. Thus an intron named "5" is acceptable when used within its scope. Until there are scope rules in a database, such simple names cannot be used reliably [21].

The "note" in the current GenBank implementation does not satisfy the naming requirements because the data contained within it cannot be obtained precisely with a single straight forward algorithm. *The use of notes for naming objects should be completely eliminated. All such notes should be replaced with appropriate biological names.*

3. **Every object which represents a change of the sequence has that change recorded in a computer manipulatable format.** Without a precise algorithm for how to change the object, programs which perform large statistical analysis of the database cannot be built. Once again, "note" fails to satisfy this requirement.

   A single scheme can be used to define insertions, deletions and base changes. A set $\{L, R, S\}$ consisting of two base positions (L, left and R, right, integers with $L < R$) and a "substitution sequence" (S) is defined for each mutation. The operation to generate the mutant sequence has two steps. First, the sequence between but not including L and R is deleted. Second, the substitution sequence S is placed between the original positions corresponding to L and R. Examples:

   {3, 4, 'A'} inserts an A between positions 3 and 4;

   {3, 5, 'ϕ' } deletes base 4. ϕ is a symbol designating an empty sequence;

   {3, 5, 'G'} replaces base 4 with a G.

   {3, 20, 'GTGC'} replaces bases 4 through 19 with GTGC.

   L and R may have values outside the bounds of the sequence being modified to allow for deletions or attachment of new sequence to one or the other end of a sequence. It is possible for S to point to a sequence elsewhere in the database, by giving the full name of the other sequence. This would allow the creation of artificial constructs.

4. **No object is ever duplicated.** Duplication in the primary database often leads to inconsistency when one of the objects is subsequently corrected but the other is not. It also wastes space. Duplicated objects bias massive statistical analyses of sequences, and may invalidate their conclusions. This is an example of Principle 1.

5. **Every object ALWAYS has a machine parsable record of the evidence supporting it.** Evidence for sequence objects falls into two categories: those determined by experimental data (such as footprinting, S1 mapping, etc.) and those determined by a computer program using only sequence data. In both cases a list of method names and references must be defined. For programs it is important to record the algorithm, the program name, and the version number and a reference. Computer programs that search sequences embody models about what is in sequences and are therefore subjective. Consensus sequences should never be used because they are an extremely poor model [13]. Objects located by their sequence patterns are effectively unsubstantiated hypotheses and should always be considered tentative. They should be removed when experimental data supporting or refuting them are published. Computer search results are useful as predictions, but unless they are identifiable, they interfere with statistical analysis of the database by contaminating data sets. A cleaner solution would be to disallow any objects that do not have experimental evidence.

6. **Every object always has one or more references.** These allow one to locate and repeat the original experimental data or program run. The ability to do this would allow anybody to check the database for errors and fraud.

7. **Every object has a natural location.** Ultimately, this is given by a chromosome name and positions on the chromosome. Despite the current trend, binding sites are not discrete box-like objects defined by a consensus sequence. This is most clearly demonstrated by sequence logos [39, 11]. It is generally misleading to record two outer points as the edges of a binding site because the site may extend well beyond the conventionally accepted consensus sequence or "box" [13]. *A consensus or "box" is merely a poor model of a binding site, not the site itself.* Because determining the boundary should be the subject of careful and continuing experimental, information theory or statistical analysis, binding site locations should only be recorded as a "zero position" and an orientation. They may be either asymmetric or symmetric. Asymmetric sites require orientation information, but dyadic symmetry sites do not. Symmetric sites may be "odd", in which case there is a central base and so there are an odd number of bases no matter what the extent of the site is. Only the position of the central base needs to be recorded. By our convention [11], this base is the zero position because it is convenient to label it 0 on aligned listings of odd or asymmetric sites extracted from a database (Fig. 1). Symmetric sites may also be "even" and therefore lack a central base. In this case the zero position to be recorded in a database is between two bases, having a position of $i + \frac{1}{2}$, where i is an integer. For the purposes of producing aligned listings

or sequence logos of even binding sites extracted from a database, we set $i = 0$, so that the center of the symmetry lies between bases 0 and 1.

8. **Whenever possible, objects should be constructed by using the unique names of smaller objects.** For example, exons may end at a splice donor site, the end of the transcript, the end of the actual sequence or the end of the known sequence. Except for the last these are not distinguished in GenBank, so it is impossible to automatically list all true donor sites by using the end of the exon. Introns, exons, and other long sequence objects should be recorded by naming two other objects such as the donor and acceptor sites. *These binding sites are the actual objects recognized by the cellular machinery so they are the fundamental data to be recorded.* Naming them allows direct automatic access to the individual sites, in addition to the defined regions.

Sequence Piece Definition: A sequence (or "piece") is a series of nucleic-acid or amino acids represented by alphabetic symbols. Alignment gaps are allowed, and are symbolized by a dash (-). Each sequence has associated with it:

1. **A type**. This indicates the kind of molecule being described. It may be DNA, RNA or protein. It is useful to allow definition of alphabetic (a to z), numeric, and symbolic vectors also, as this allows sequence logos to be defined in subsets of the database [39].

2. **A number of strands**. Typical molecules are single or double stranded, but higher numbers are possible.

3. **A topology.** The molecule may be linear, circular or repeated. Branched sugars or RNAs will also have to be defined eventually.

4. **A coordinate system.** This follows from Principle 3 because when one is working with a partial fragment of a sequence, it is useful to have the original numbering system maintained in order to compare results output from different programs. The Delila System does this [6]. For example, suppose one wanted to determine RNA folded structures for a series of mutant sequences. A flexible coordinate system would maintain most of the numbering despite small insertions or deletions. This would facilitate comparison of the different folds. By Principle 3, the original database should also have a coordinate system capable of handling complex coordinate systems. When a single sequence is derived from several other sequences, each portion may have its own coordinates. Type, topology and coordinates may be listed compactly in the form:

   DNA ds C(1 100)(120 150)(15 1)(150 200)

   This indicates that a double stranded (ds) circular (C) DNA sequence is being described. The sequence, given $5'$ to $3'$, starts at base 1 and proceeds to 100. There is a gap in the numbering (*e.g.* from a deletion or a chimeric construction), followed by a segment numbered from 120 up to 150, then an inserted segment numbered from 15 *down to* 1. Finally the circle is closed with sequences running from 150 to 200. With this method, mutant sequences can be created which still have nearly the same numbering as the original sequence. This facilitates comparison between sequences. In this scheme, coordinates of each base must be defined with two numbers, 2@135 could be a notation to indicate the second set of sequence, base 135 of the example given above.

In nature no two chromosomes are the same, so how can we have a coordinate system? In any particular case we have a specific sequence. It is technically easy to store a canonical sequence and then to store instructions for creating all the polymorphisms and variations observed in nature. Features could be automatically moved to any specific strain sequence created on the fly.

5. **An alignment.** When a set of sequences are extracted and aligned, a single base must be designated as the aligned base. By Principle 3, this subset of the database must be in the same form as the original database. The first base of the sequence can be the default alignment point. The primary database would not have alignment gaps.

6. **A species designation list.** More than one species may be required to describe artificial constructions. In this case each coordinate segment must be identified.

7. **A map location**, in standard genetic coordinates. This includes the orientation of the sequence relative to the chromosome (if known).

8. **A list of associated objects** for the sequence. Every object is associated with its originating species (or inorganic synthesis), but there can be a large number of objects on a nucleic acid. To allow programs full flexibility, these objects must include not only introns, exons and coding regions, but also the known points of protein modification, crosslinking and three dimensional structure (Principle 8). This complete viewpoint allows one (for example) to directly extract a subset of the database and then to compare protein structure to intron structure.

9. **References**. A list of references for the sequence and objects.

Reference Definition: A reference is a citation to the primary scientific literature. The reference must include: authors, title, journal, volume, pages, year. The name, address, phone, email address and other identifying information of the originator should also be stored when it is available. Because this kind of extra information would allow more rapid use of the database and would tend to keep authors responsible for upgrading their data, it should be made available to the public as part of the database. For implementation, a practical modern format for these data is the BiBTeX format since it can be parsed by machine and allows direct typesetting [42].

CONCLUSION

A major goal for a genetic sequence database is to allow us to easily isolate specific sequences for analysis. We would like to be able to write:

```
species "Homo sapiens"; (* define the species *)
strain "J. D. Watson";  (* define the strain *)
maternal chromosome;    (* define the chromosome set *)
locus gene ADH;         (* a locus whose type is gene, named ADH *)
locus exon 3;           (* a sub-locus of the ADH locus *)
get all locus;          (* by implication the last defined sub-locus would be used *)
```

and obtain the sequence. This is not possible now for many reasons described in this paper. To support instructions like this, we should store named and typed objects with as few absolute coordinates as possible and the data should be entirely machine parsable:

```
locus   type              name   location
-----   ----              ----   --------
agene   cap               c      2456 orientation: +
agene   splice-donor      d      4595 orientation: +
agene   splice-acceptor   a      4896 orientation: +
agene   polyA             p      5041 orientation: +
agene   exon              1      cap c, splice-donor d - 1
agene   intron            1      splice-donor d, splice-acceptor a
agene   exon              2      splice-acceptor a + 1, polyA p
agene   mRNA              m      exon 1, exon 2
```

The problems described here led me to propose a basic philosophy and a design for the database. Discussing fundamental design issues, rather than the nitty gritty of implementation, may seem to be many years behind the current status of the databases, but we must begin to have this level of discourse if the databases are ever to become well designed, comprehensible, stable, and more scientifically valuable. The task before us is monumental. Every molecular biologist has a stake in the future form of the database and we can each play an important role in helping to achieve a richly annotated but clean database.

ACKNOWLEDGEMENTS

I thank Brian Fristensky for suggesting the priority principle for GenBank publication; David George, for his rewording on Principles 4 and 5, and for recognizing the relationship between Principles 1 and 7; Pete Rogan for suggesting the editorial review system in Principle 4 and the medical example discussed in Principle 6; and Sue Aldor for wording in Principle 8. Many useful discussions of these topics on the internet news groups bionet.general, bionet.molbio.bio-matrix and bionet.molbio.genbank would not have been possible without the support of the bionet news groups by David Kristofferson [43]. I thank David Lipman for suggesting the "taxonomy" of errors. Stacy Bartram discovered the duplicated accession numbers. I thank Michael J. Cinkosky, Kirill Degtiarenko, Paul N. Hengen, Ingrid Jakobsen, David Lipman, Maureen Madden, Jake Maizel, Jim Ostell, Peter Rogan, Denise Rubens, Kenn Rudd, and Bruce Shapiro for useful discussions and insightful comments on the manuscript.

REFERENCES

1. Jakobsen, I. (March 25, 1993) Genbank entries from journal scanning, bionet.molbio.genbank, Human Genetics Group, John Curtin School of Medical Research, Australian National University, ingrid@helios.anu.edu.au.

2. George, D. (July 10, 1992) RE: A Philosophy for GenBank ... and Others, bionet.general, Protein Identification Resource (PIR), National Biomedical Research Foundation, Georgetown University Medical Center, george@nbrf.georgetown.edu.

3. Martin, J. (1977) Computer Database Organization, Prentice-Hall, Inc., Englewood Cliffs, NJ second edition.

4. Cole, S. P. C. and Deeley, R. G. (1993) *Science* **260**, 879–879.

5. Schneider, T. D. (1980) World Wide Web Universal Record Locator: ftp://ftp.ncifcrf.gov/pub/delila/libdef.Z.

6. Schneider, T. D., Stormo, G. D., Haemer, J. S., and Gold, L. (1982) *Nucl. Acids Res.* **10**, 3013–3024.

7. Schneider, T. D., Stormo, G. D., Yarus, M. A., and Gold, L. (1984) *Nucl. Acids Res.* **12**, 129–140.

8. Stormo, G. D., Schneider, T. D., and Gold, L. M. (1982) *Nucl. Acids Res.* **10**, 2971–2996.

9. Stormo, G. D., Schneider, T. D., Gold, L., and Ehrenfeucht, A. (1982) *Nucl. Acids Res.* **10**, 2997–3011.

10. Schneider, T. D., Stormo, G. D., Gold, L., and Ehrenfeucht, A. (1986) *J. Mol. Biol.* **188**, 415–431.

11. Papp, P. P., Chattoraj, D. K., and Schneider, T. D. (1993) *J. Mol. Biol.* **233**, 219–230.

12. Schneider, T. D. (1994) *Nanotechnology* **5**(1), 1–18.

13. Stephens, R. M. and Schneider, T. D. (1992) *J. Mol. Biol.* **228**, 1124–1136.

14. Cuticchia, A. J., Chipperfield, M. A., Porter, C. J., Kearns, W., and Pearson, P. L. (1993) *Science* **262**, 47–48.

15. Benson, D., Lipman, D. J., and Ostell, J. (1993) *Nucl. Acids Res.* **21**, 2963–2965.

16. A. H. Soukhanov and K. Ellis, (ed.) (1984) Webster's II New Riverside University Dictionary, Houghton Mifflin Co., Boston, MA.

17. Watson, J. D. (1990) *Science* **248**, 44–49.

18. Cantor, C. R. (1990) *Science* **248**, 49–51.

19. Schroeder, J. L. and Blattner, F. R. (1982) *Nucl. Acids Res.* **10**, 69–84.

20. George, D. G., Orcutt, B. C., Mewes, H.-W., and Tsugita, A. (1993) *Protein Seq. Data. Anal.* **5**, 357–399.

21. (Feb 1, 1994) The DDBJ/EMBL/GenBank Feature Table: Definition, Version 1.06, World Wide Web, Universal Record Locator: ftp://ncbi.nlm.nih.gov/ncbi-genbank/docs.

22. Brunak, S., Engelbrecht, J., and Knudsen, S. (1990) *Nature* **343**, 123.

23. Lamperti, E. D., Kittelberger, J. M., Smith, T. F., and Villa-Komaroff, L. (1992) *Nucl. Acids Res.* **20**, 2741–2747.

24. Lopez, R., Kristensen, T., and Prydz, H. (1992) *Nature* **355**, 211.

25. Binns, M. (1993) *Nucl. Acids Res.* **21**, 779.

26. Lisser, S. and Margalit, H. (1993) *Nucl. Acids Res.* **21**, 1507–1516.

27. Saracevic, T. and Kesselman, M. (1958) In G. T. Tzotzos, (ed.), Biotechnology R&D Trends: Science Policy for Development, New York, New York: The New York Academy of Sciences. pp. 135–144.

28. Lawrence, C. B. and Solovyev, V. V. (1994) *Nucl. Acids Res.* **22**, 1272–1280.

29. Ostell, J. The NCBI Software Development ToolKit. Version 2.0, (1993).

30. Sutcliffe, J. G. (1979) *Cold Spring Harb. Symp. Quant. Biol.* **43**, 77–90.

31. Oliver, S. G., van der Aart, Q. J. M., Agostini-Carbone, M. L., Aigle, M., Alberghina, L., Alexandraki, D., Antoine, G., Anwar, R., Ballesta, J. P. G., Benit, P., Berben, G., Bergantino, E., Biteau, N., Bolle, P. A., Bolotin-Fukuhara, M., Brown, A., Brown, A. J. P., Buhler, J. M., Carcano, C., Carignani, G., Cederberg, H., Chanet, R., Contreras, R., Crouzet, M., Daignan-Fornier, B., Defoor, E., Delgado, M., Demolder, J., Doira, C., Dubois, E., Dujon, B., Dusterhoft, A., Erdmann, D., Esteban, M., Fabre, F., Fairhead, C., Faye, G., Feldmann, H., Fiers, W., Francinques-Gaillard, M. C., Franco, L., Frontali, L., Fukuhara, H., Fuller, L. J., Galland, P., Gent, M. E., Gigot, D., Gilliquet, V., Glansdorff, N., Goffeau, A., Grenson, M., Grisanti, P., Grivell, L. A., de Haan, M., Haasemann, M., Hatat, D., Hoenicka, J., Hegemann, J., Herbert, C. J., Hilger, F., Hohmann, S., Hollenberg, C. P., Huse, K., Iborra, F., Indige, J., Isono, K., Jacq, C., Jacquet, M., James, C. M., Jauniaux, J. C., Jia, Y., Jimenez, A., Kelly, A., Kleinhans, U., Kreisl, P., Lanfranchi, G., Lewis, C., van der Linden, C. G., Lucchini, G., Lutzenkirchen, K., Maat, M. J., Mallet, L., Mannhaupt, G., Martegani, E., Mathieu, A., Maurer, C. T. C., McConnell, D., McKee, R. A., Messenguy, F., Mewes, H. W., Molemans, F., Montague, M. A., Falconi, M. M., Navas, L., Newion, C. S., Noone, D., Pallier, C., Panzeri, L., Pearson, B. M., Perea, J., Philippsen, P., Pierard, A., Planta, R. J., Plevani, P., Poetsch, B., Pohl, F., Purnelle, B., Rad, M. R., Rasmussen, S. W., Raynal, A., Remacha, M., Richterich, P., Roberts, A. B., Rodriguez, F., Sanz, E., Schaaff-Gerstenschlager, I., Scherens, B., Schweitzer, B., Shu, Y., Skala, J., Slonimski, P. P., Sor, F., Soustelle, C., Spiegelberg, R., Stateva, L. I., Steensma, H. Y., Steiner, S., Thierry, A., Thireos, G.,

Tzermia, M., Urrestarazu, L. A., Valle, G., Vetter, L., van Vliet-Reedijk, J. C., Voet, M., Volckaert, G., Vreken, P., Wang, H., Warmington, J. R., von Wettstein, D., Wicksteed, B. L., Wilson, C., Wurst, H., Xu, G., Yoshikawa, A., Zimmermann, F. K., and Sgouros, J. G. (1992) *Nature* **357**, 38–46.

32. Naur, P., Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., and Woodger, M. (January, 1963) *Comm. ACM* **6**(1), 1–17.

33. Roberts, L. (1993) *Science* **262**, 504–505.

34. Cinkosky, M. J., Fickett, J. W., Gilna, P., and Burks, C. (1991) *Science* **252**, 1273–1277.

35. Fristensky, B. (July 6, 1992) Re: missing accession numbers, bionet.molbio.genbank, University of Manitoba, Winnipeg, Manitoba, Canada, frist@ccu.umanitoba.ca.

36. Schneider, T. D. (1991) *J. Theor. Biol.* **148**, 83–123.

37. Schneider, T. D. (1991) *J. Theor. Biol.* **148**, 125–137.

38. Aldhous, P. (1993) *Science* **262**, 502–503.

39. Schneider, T. D. and Stephens, R. M. (1990) *Nucl. Acids Res.* **18**, 6097–6100.

40. Cuticchia, A. J., Fasman, K. H., Kingsbury, D. T., Robins, R. J., and Pearson, P. L. (1993) *Nucl. Acids Res.* **21**, 3003–3006 (The Online Mendelian Inheritance In Man Gene Map is at World Wide Web Universal Record Locator: http://gdbwww.gdb.org/omimdoc/omimtop.html).

41. Jensen, K. and Wirth, N. (1975) Pascal User Manual and Report, Springer-Verlag, New York.

42. Lamport, L. (1986) LaTeX: A Document Preparation System, User's Guide & Reference Manual, Addison-Wesley Publishing Company, Reading, Massachusetts.

43. Bleasby, A., Griffiths, P., Harper, R., Hines, D., Hoover, K., Kristofferson, D., Marshall, S., O'Reilly, N., and Sundvall, M. (1992) *Nucl. Acids Res.* **20**, 4127–4128.

Figure 1: Aligned listing of incomplete human splice acceptor sites.
The sequences are a part of the data set described in reference [13]. The LOCUS name (some of which no longer exist in the database) is followed by the alignment or "zero" position (L) and the sequence. Except for the last two sequences, L is also the length of sequence given on the intron side. The set of numbers above the sequences are the positions (relative to the alignment position) written vertically. The data shown is all of the data provided by the authors in GenBank database number 62 (December 1989); dashes indicate missing data. The edge of available sequence data would rarely correlate perfectly with multiples of 5, so we conclude that different authors arbitrarily decided what was important, and did not report all the data which they have. Our statistical analysis showed that the acceptor site extends from position +2 to *at least* position -25, and perhaps as far as -30 and beyond [13]. Thus the blocks of intentionally missing data affect statistical studies of these sequences.

Figure 2: Overview of a "natural" organization for sequence data.
The database (rectangle) consists of several species (ellipses). Each species contains one or more chromosomes (circles c1, c2, and c3). Individual sequenced pieces of nucleic acid are stored within the chromosome data objects (line segments or squiggles p1, p2 and p3). Strain information could be stored either at the species level (which is likely to be inefficient in storage because it is redundant) or as specific changes at the sequence level (which would be storage-efficient and could use the same format as mutations and sequence conflicts).

Figure 3: Detail of species 2, chromosome 1 in Fig. 2.
The region separating sequence piece 1 (line segment p1) and piece 2 (p2) is dashed to indicate that it is not sequenced. Three genetic loci (rectangles) are defined, each containing several sequence objects (arrows o1, o2, and o3). Note that the loci and objects may overlap without causing difficulty in the scope of names, so long as the names and the extents of loci and objects are chosen carefully. For example, locus 2 object 1 (in the unsequenced region) is distinct from locus 3 object 1 (which has been sequenced), while in locus 1, object 3 overlaps object 2. Object orientation is indicated by the direction of the arrows. Objects and loci can span across disconnected pieces.

```
                            --------------------------
                            3333332222222222111111111---------- ++
                            54321098765432109876543210987654321012
LOCUS            L          ......................................

HUMCYP4A3        5          -------------------------------ctcagaa
HUMCYP4A4        5          -------------------------------ttcagga
HUMCYP4A5        5          -------------------------------tacagac
HUMCYP4A6        5          -------------------------------ctcagca
HUMCYP4A7        5          -------------------------------tgcagag

HUMAMY105       10          --------------------------tttcttctaggt
HUMAMY106       10          --------------------------tcaaaaatagga
HUMAMY107       10          --------------------------taactttcaggc
HUMAMY108       10          --------------------------tgtaattaagga
HUMAMY110       10          --------------------------tattttacagga

HUMBHA02        15          ---------------------accctgtcttcctaggg
HUMBHA03        15          ---------------------tttctgtcatttcagat
HUMBHA04        15          ---------------------tctacatcttcctaggt
HUMBHA05        15          ---------------------gtttgttctgcacagtt
HUMBHA06        15          ---------------------cactttaacctacagga
HUMBHA07        15          ---------------------tcttgtgcctttcaggg
HUMBHA08        15          ---------------------tttctcttggcttaggt
HUMBHA09        15          ---------------------tctcttgggattcagga
HUMBHA10        15          ---------------------tcctctcctctccaggc
HUMBHA11        15          ---------------------gaccttttataacagat
HUMBHA12        15          ---------------------cttgttttccctcaggt
HUMBHA13        15          ---------------------ccccttttcctccaggc
HUMBHA14        15          ---------------------catgtcctcttgcaggc

HUMARG2         20          ----------------ctttattttttaatgttcagcc
HUMARG3         20          ----------------tcaaaactttttaattttagag
HUMARG4         20          ----------------caaaattttttccccaaaagtt
HUMARG5         20          ----------------tgaaaacattgtaattttagat
HUMARG6         20          ----------------tcttaatttctcttttatagct
HUMARG7         20          ----------------cctttcccacttcttaaaagaa
HUMARG8         20          ----------------attacaatttgttgttgtaggg

HUMAS02         25          -----------ctgcagagtagctctgcttttgcagag
HUMAS03         25          -----------caggttgttcctcgactcccgccagac
HUMAS04         25          -----------gagcctctccgcttctgcttctcaggc
HUMAS07         25          -----------ttcgccgctttctgtcttttttcagaa

HUMATPCAPM     129          atgatttatttctaagttcattcccctgtgttgtagct
HUMATPCAPM     684          gctgagcaagctgtcacaatctctgattccttgcagat
```

Figure 1

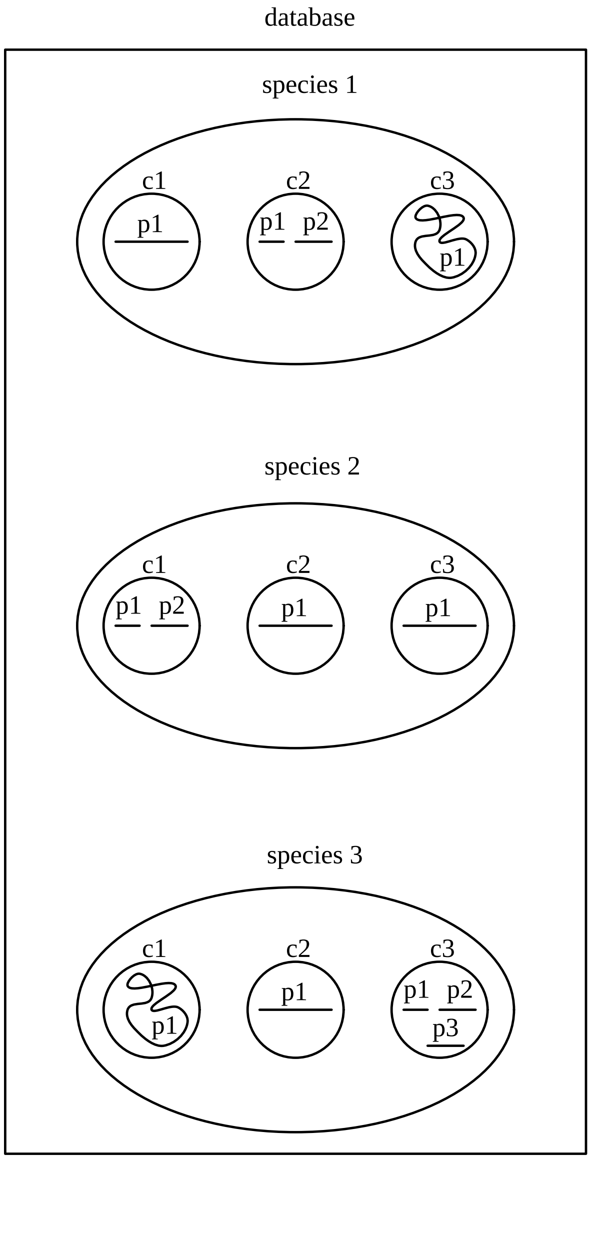
database
species 1
c1
p1
c2
p1 p2
c3
p1
species 2
c1
p1 p2
c2
p1
c3
p1
species 3
c1
p1
c2
p1
c3
p1 p2
p3

Figure 2

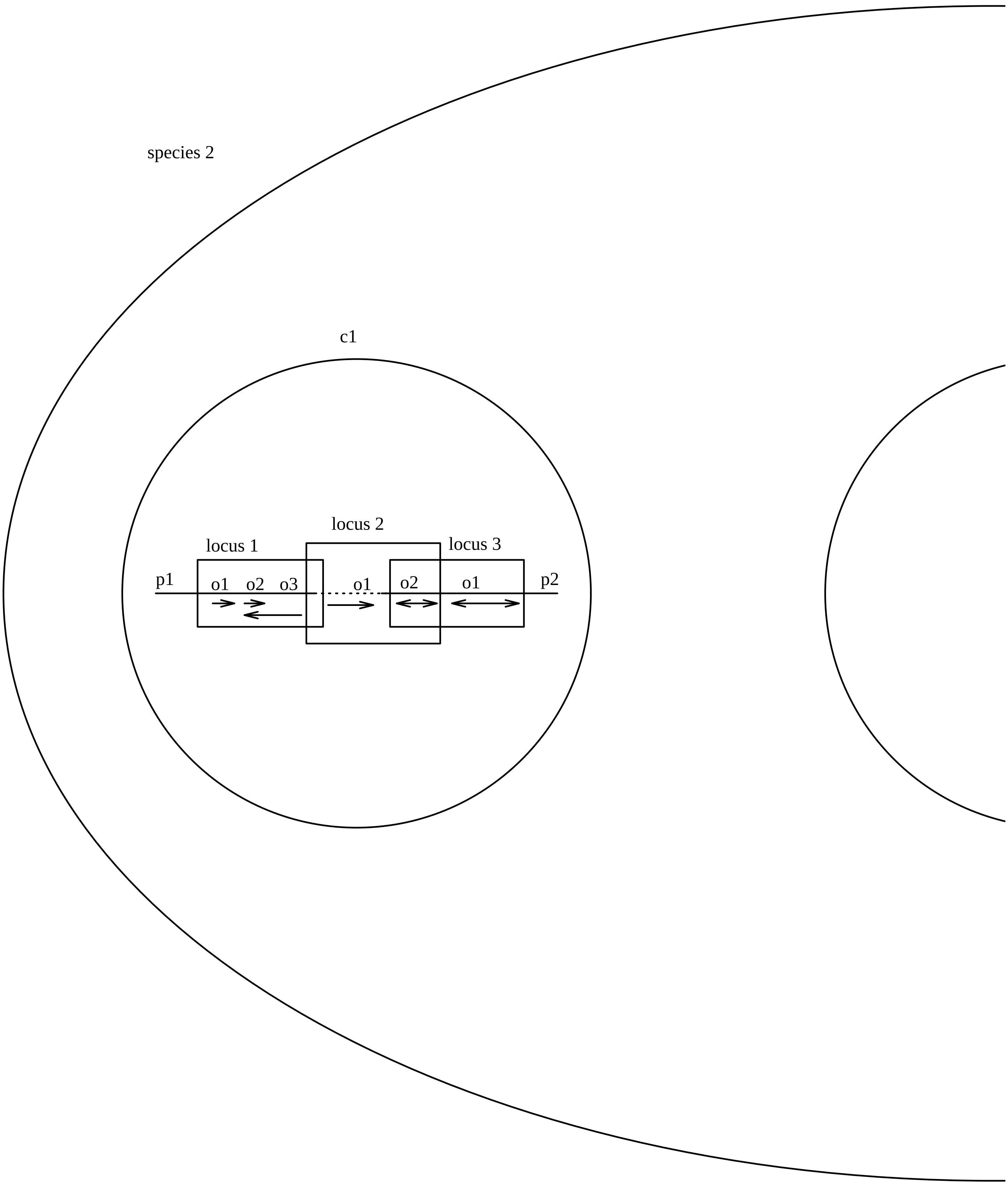

Figure 3