© *1993 Oxford University Press* *Nucleic Acids Research, 1993, Vol. 21, No. 25* **5997-6003**

Feature expressions: creating and manipulating sequence datasets

Brian Fristensky
Department of Plant Science, University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada, frist@cc.umanitoba.ca

Received August 16, 1993; Revised and Accepted November12, 1993

ABSTRACT

Annotation of features, such as introns, exons and protein coding regions in GenBank/EMBL/DDBJ entries is now standardized through use of the Features Table (FT) language. The essence of the FT language is described by the relation *'expression → sequence'* meaning that each FT expression evaluates to a sequence. For example, the expression M74750:1..50 evaluates to the first 50 bases of the sequence with accession number M74750. Because FT is intrinsic to the database definition , it can serve as a software- and platform-independent *lingua franca* for sequence manipulation. The XYLEM package makes it possible to create and manipulate sequence datasets using FT expressions. FEATURES is a program that resolves FT expressions into their corresponding sequences. Annotated features can be retrieved either by feature key or by expression. Even unannotated portions of a sequence can be retrieved by user-generated FT expressions. Applications of the FT language include retrieval of subsequences from large sequence entries, generation of chromosome models or artificial DNA constructs, and representation of restriction maps or mutants.

INTRODUCTION

While the widespread availability of sequence databases has been of great value to molecular biologists, most database usage is limited to a few simple tasks: searching for entries by keyword, retrieval of entries, and sequence similarity searches. More sophisticated projects often require the creation of large database subsets, representing particular taxa, organs, tissues, or other groupings which merit comparison. One of the earliest studies of this type analyzed 124 mRNA sequences from *E.coli* to infer a set of rules for identification of ribosome binding sites [1]. More recently, 369 Alu1 dispersed repetitive elements were categorized into subfamilies to enable reconstruction of their evolutionary history [2]. Such projects require not only the ability to organize sequences into discrete groups, but also to extract specific subsequences from each database entry for analysis of comparable features.

A sequence query language, that is, a language in which expressions, upon evaluation, yield sequence, would offer many advantages in dataset construction. The sequences themselves need not be stored, but rather, the instructions necessary to recreate the dataset. Interestingly, the most ambitious attempts at writing sequence query languages predate GenBank [3] itself. Schroeder and Blattner [4] described DNA*, which permitted concatenation and complementation of DNA sequences using a terse syntax. Another approach was that of DELILA (DEoxyribonucleic acid LIbrary LAnguage, [5]. DELILA encompassed both a hierarchical syntax for description of genomes, as well as a query language in which named features served as reference points within a coordinate system. Because both languages predated the current databases, they do not contain syntax for reference to database entries. While more recent tools have been able to parse GenBank entries for direct use of data fields by other programs [6], automated access to the features annotated in the Features Table has been difficult to realize.

The development of the Feature Table language (FT) [7] as an integral part of database annotation was a fundamental step in making sequence data more useable because each feature in a GenBank entry is now annotated in a standard, machineparsable syntax. The universality of this language now makes it possible to specify any DNA sequence using an expression, as given by the relation

expression ⟶ *sequence*

This task has been implemented in the FEATURES program, which is part the XYLEM package, to be described in this paper (Table I). While fully accessible through a menu-driven interface, the simplest form of the FEATURES command is

features *expression* > *sequence*

meaning that FEATURES can take a FT expression as input and write a sequence to the output. For example, given the following feature annotated in the GenBank entry with primary accession number M74750:

```
terminator      609..650
                 /label=T7-terminator
```

typing the command

```
features M74750:T7-terminator
```

would return the sequence

```
ataacccttggggcctctaaacgggtcttgaggggttttt
```

representing that part of the sequence spanning bases 609 to 650, as identified by the field 'label=T7-terminator'.

Table I. List of XYLEM programs and functions.

Program	Function
High-level tools	
FINDKEY	Search for one or more keywords in database
FETCH	Retrieve one or more entries from database
FEATURES	Extract features by feature key or expression
Low-level tools	
SPLITDB	Split a database into annotation, sequence and index
IDENTIFY	Used by FINDKEY to identify entries containing keywords
GETLOC	Used by FETCH to retrieve entries from a split database
GETOB	Used by FEATURES to parse Feature Table expressions
UDS	Update an existing dataset with new versions of entries
DBSTAT	Calculate amino acid frequencies in a protein database
RIBOSOME	Translate file of nucleic acid sequences into protein
SHUFFLE	Given a random seed, shuffles each sequence in a file
REFORM	Multiple alignment printing tool
GBUPDATE	Download GenBank database by FTP; calls SPLITDB
PIRUPDATE	Download PIR database by FTP; calls SPLITDB

The XYLEM tools (Table I) automate the management of online databases, as well as the construction of sequence database subsets. Even non-expert users should be able to create datasets for use in multiple alignments, phylogenetic studies, structure comparisons and other types of analyses.

METHODS

Program organization

All programs are written in Pascal using SUN Pascal 2.0, or as Unix c-shell scripts. Presently, both Sparc binaries as well as Pascal source code are available, and a version in C is being prepared. The use of shell scripts has both advantages and disadvantages. Shell scripts are easier to modify to suit local needs than are programs in a compiled language. Also, system-dependent code has generally been relegated to the shell scripts, while the Pascal programs adhere strictly to the Standard [8], and should need little or no modification regardless of the compiler used.

The XYLEM tools work with databases formatted as illustrated in Figure 1. SPLITDB splits a file of GenBank, EMBL, PIR or VecBase entries into annotation (.ano), sequence (.wrp), and an index (.ind) listing the location of each entry in the annotation and sequence files. Splitting annotation from sequence speeds both keyword searches in annotation and global sequence similarity searches. Sequence files (.wrp) produced by SPLITDB are in the format required for the FASTA programs of W. R. Pearson [9].

XYLEM has been designed on the principle that one or more lower-level tools can be incorporated into higher-lever tools, creating new functionalities while at the same time preserving the ability to use the low-level tools directly (Figure 1). All programs can be executed by supplying parameters and filenames at the command line. As well, the higher-level programs FINDKEY, FETCH and FEATURES can be used interactively, either as text only applications, or as menu items in the Genetic Data Environment (GDE) [10].

Compatibility with other databases

Although initially written to work with GenBank, FETCH and FINDKEY, along with the programs they call, can also be used with the PIR [11] and Vecbase [12] databases. Adaptation for use of FEATURES with the EMBL database [13] is in progress.

Compatibility of FEATURES with other database software

In recognition of the fact that local copies of GenBank may be formatted for other software packages, FEATURES has been designed to operate independently of database format. This is possible because the process of feature extraction consists of two steps: retrieval of entries and evaluation of expressions. The retrieval step is implemented in FEATURES as a call to FETCH in the form '**fetch** *accession-file outputfile*' where *accession-file* is a list of accession numbers, and the corresponding entries are written to *outputfile* in standard GenBank flatfile format. <u>Consequently, FEATURES can work with any software that, given a list of accession numbers, retrieves GenBank flatfile entries.</u> All that is required is to replace the two lines that call FETCH with the corresponding command for one's local system. In principle, the sequence retrieval step could even be implemented for retrieval of sequences from the NCBI email server, although such a scheme might be awkward to implement reliably.

Availability

All programs are available free of charge. They may be obtained through anonymous FTP at ftp.cc.umanitoba.ca (130.179.16.24) in the directory psgendb. The SUN4 distribution may also be obtained by sending the author one 1.44Mb HD diskette or a 1/4 in. tape, which will be written in tar format. Users wishing distribution on other media or for other platforms please inquire.

IMPLEMENTATION

Creation of customized datasets

Datasets containing groups of sequences for comparison can be created in a three step process: a) identification of entries by keyword search b) retrieval of hits c) extraction of the appropriate sequence fragments. As shown in Fig. 2-a, the FINDKEY menu allows the user to either type in a single keyword or create a list of one or more keywords. In the example, a user wishing to find entries containing the keyword 'chitinase' has chosen to search the plant division of GenBank. (Alternatively, a user-created dataset could have been searched.) Two files are generated: a list of annotation lines found (chitinase@pln.fnd), as well as a list of GenBank LOCUS names (chitinase@pln.nam) corresponding to the hits. When executed from GDE, these files would appear in textedit windows.

Fundamental to the power of XYLEM is the ability to perform operations on namefiles, which can function as virtual datasets. Unix commands such as "comm" or "diff" can be used to find the logical union, intersection or difference between two namefiles. Real datasets (ie. files of entries) can be generated from virtual datasets (ie. namefiles) using FETCH. In Fig. 2-b, the sequences represented in the virtual dataset created in Fig. 2-a are retrieved directly to the GDE sequence alignment editor.

The versatility of FETCH is illustrated in the menu. Single names or accession numbers can be typed in the menu, or a list of names or accession numbers read from a file. Annotation only, sequence only, or both can be retrieved. Entries can be retrieved either from the original databases, or from user-created datasets. When FETCH is run from GDE, output can go directly to the sequence editor, to a textedit window, or to a file.

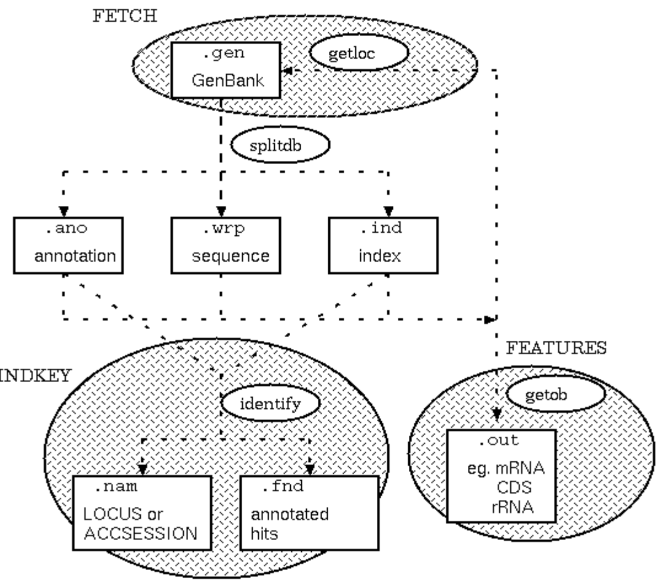


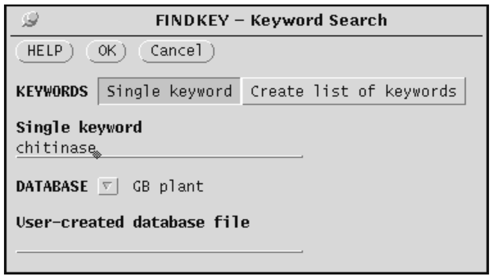
Figure 1. XYLEM Schematic. Dashed lines describe the flow of data (rectangles) between programs (ovals). The three shell scripts, FINDKEY, FETCH and FEATURES (shaded ovals) each consist of calls to low level programs and Unix commands. While not illustrated here, FEATURES calls FETCH to retrieve sequences required to resolve FT expressions. FINDKEY and FETCH require that the dataset be divided into annotation, sequence and index by SPLITDB. FEATURES calls SPLITDB automatically, if needed.

Feature extraction using feature key or FT expression

Figure 2-c is an example of extraction by feature key. The FEATURES menu has been set to extract sequence from GenBank entry EPFCPCG (M81884), the plastid genome from the epiphytic plant *Epifagus virginiana*. Note that in this example, a file containing a single entry (EPFCPCG.gen) is used for input. Alternatively, data could have been directly read from GenBank. The user has specified that all regions annotated with the 'tRNA' feature key are to be extracted. FEATURES creates a message file, a sequence file and an expression file resulting from the extraction of the 28 tRNAs annotated in this entry. At the top of Figure 3 is the Feature Table annotation for the first tRNA in this entry, and the resultant output files, written to textedit windows by GDE, below. The message (.msg) file is a log of feature extraction, in which the expression is re-written as it is evaluated, along with any qualifier lines accompanying the feature. The resultant sequence for the tRNA is written to the sequence file (.out), in a format compatible with the Pearson FASTA programs [9]. As the sequence file is generated, an expression file (.exp) containing expressions in place of sequence is also produced.

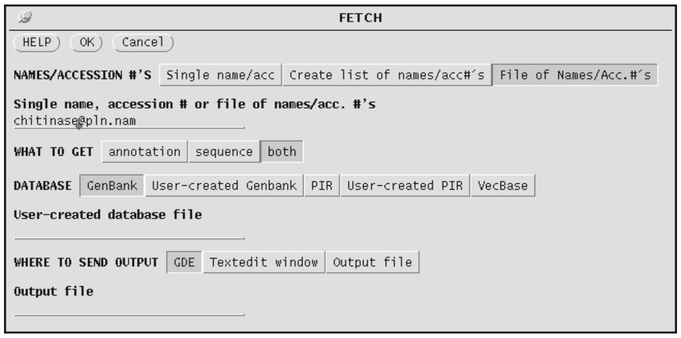
The expression file facilitates a second means of feature extraction, namely, extraction by expression. Given the expression file as input, FEATURES can re-generate the sequence file by replacing expressions with sequence. This capability helps automate the creation and maintenance of customized datasets. One can choose some subset of features to be evaluated, rather than all features bearing a particular feature key. For example, to generate <u>only</u> Ile-tRNAs, one could edit out all other tRNA expressions from the expression

a



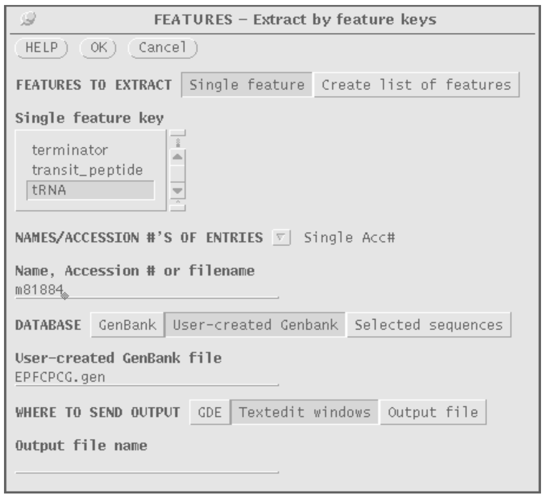
KEYWORDS [✓ Single keyword] Create list of keywords
Single keyword
chitinase
DATABASE GB plant
User-created database file

b



NAMES/ACCESSION #'S Single name/acc Create list of names/acc#'s [✓ File of Names/Acc.#'s]
Single name, accession # or file of names/acc. #'s
chitinase@pln.nam
WHAT TO GET annotation sequence [✓ both]
DATABASE [✓ GenBank] User-created Genbank PIR User-created PIR VecBase
User-created database file
WHERE TO SEND OUTPUT [✓ GDE] Textedit window Output file
Output file

c



FEATURES TO EXTRACT [✓ Single feature] Create list of features
Single feature key
terminator
transit_peptide
[✓ tRNA]
NAMES/ACCESSION #'S OF ENTRIES Single Acc#
Name, Accession # or filename
m81884
DATABASE GenBank [✓ User-created Genbank] Selected sequences
User-created GenBank file
EPFCPCG.gen
WHERE TO SEND OUTPUT GDE [✓ Textedit windows] Output file
Output file name

Figure 2. XYLEM menus using the Genetic Data Environment (GDE) under SUN OpenWindows: a) FINDKEY, b) FETCH and c) FEATURES. A separate FEATURES menu for extraction of features by expression is also available (not shown).

file, using the qualifier fields in the message file as a guide. The expression file could now be used in the FEATURES program to generate the three Ile-tRNAs found in the plastid genome.

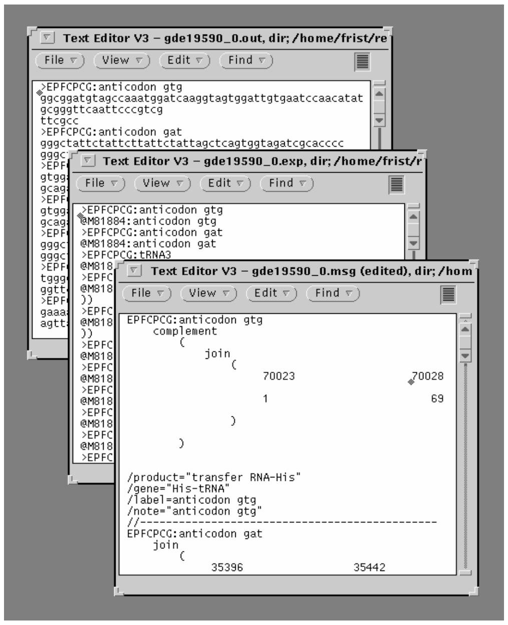
Feature expression files also allow the user to work around errors in database entries. By modifying the expression file, a feature incorrectly annotated in the database entry can still be generated correctly, pending definitive correction by database staff.

The expressions shown in Figure 3 are formulated as base

FEATURE TABLE ENTRY

```
    tRNA          complement(join(70023..70028,1..69))
                  /product="transfer RNA-His"
                  /gene="His-tRNA"
                  /label=anticodon gtg
                  /note="anticodon gtg"
```

GDE OUTPUT WINDOWS


Text Editor V3 – gde19590_0.out, dir; /home/frist/re
File
View
Edit
Find
Text Editor V3 – gde19590_0.exp, dir; /home/frist/r
File
View
Edit
Find
Text Editor V3 – gde19590_0.msg (edited), dir; /hom
File
View
Edit
Find

```
>EPFCPCG:anticodon gtg
ggcggatgtagccaaatggatcaaggtagtggattgtgaatccaacatat
gcgggttcaattcccgtcg
ttcgcc
>EPFCPCG:anticodon gat
gggctattctattcttattctattagctcagtggtagatcgcacccc
gggc
>EPF
gtgg
gcag
>EPF
gtgg
gcag
>EPF
gggc
gggc
>EPF
tggg
ggtt
>EPF
gaaa
agtt
```

```
>EPFCPCG:anticodon gtg
@M81884:anticodon gtg
>EPFCPCG:anticodon gat
@M81884:anticodon gat
>EPFCPCG:tRNA3
@M818
>EPFC
@M818
))
>EPFC
@M818
))
>EPFC
@M818
>EPFC
@M818
>EPFC
@M818
>EPFC
@M818
>EPFC
```

```
EPFCPCG:anticodon gtg
    complement
        (
            join
                (
                    70023                    70028

                    1                           69

                )

        )


/product="transfer RNA-His"
/gene="His-tRNA"
/label=anticodon gtg
/note="anticodon gtg"
//------------------------------------------------
EPFCPCG:anticodon gat
    join
        (
            35396                  35442
```

Figure 3. Sample FEATURES output windows generated using GDE. The tRNA annotated at top is one of 28 tRNAs extracted from the *E.virginiana* plastid genome using the feature key 'tRNA'. Corresponding parts of sequence (top), expression (middle) and message (bottom) files are shown. The '@' in the .exp file is used to indicate an expression, and is not part of the FT language.

FEATURE TABLE:

```
    CDS           305..640
                  /gene="flaL"
                  /label=ORF2
                  /note="flaD (sin) homologue; putative
                  /codon_start=1
```

EQUIVALENT EXPRESSIONS:

```
        M60287:305..640
        M60287:ORF2
        M60287:/label=ORF2
        M60287:/gene="flaL"
        M60287:/note="flaD (sin) homologue; putative"
```

Figure 4. A sample feature table entry and equivalent expressions. The first two expressions adhere strictly to the FT definition, while the last three are extensions allowed by GETOB.

ranges, that is, using numerical coordinates. However, the FT language provides the means for citing individual features by their 'label' qualifier fields. For example, the protein coding sequence annotated in Figure 4 could be retrieved by label, using the expression M60287:ORF2. Ideally, all features should be citable by label. However, the databases do not always create label fields for each feature. To overcome this problem, FEATURES allows a feature to be cited in any of several ways. In addition to labels and absolute base ranges, features can also be cited by qualifier lines, provided that the qualifier line used is unique within an entry. Thus the feature shown in Figure 4 could be retrieved using any of the expressions listed.

Segmented features

GenBank features may be pieced together using subsequences from other entries, a process which might be thought of as nesting of features. While it is the policy of the major databases to avoid creating entries with more than two levels of nesting, FEATURES is capable of handling any level of nesting. The entry ASYPIGG6 (M38624) describes the green visual pigment gene of the fish *Astyanax fasciatu*. Because this is a very large gene, only the exons and splice junctions have been sequenced, and the sequence split into several entries, one for each exon. Such an entry is annotated as 'SEGMENTED'. To specify the entire protein coding sequence (CDS), ASYPIGG6 contains the feature shown at the top of Figure 5.

FEATURE TABLE:

```
CDS             join(M38619:160..256,M38620:11..307,
                M38621:11..179,M38622:11..176,
                M38623:11..250,11..103)
                /product="green visual pigment"
                /gene="G101"
                /codon_start=1
```

AFTER FIRST PASS:

```
>ASYPIGG6:CDS1
@M38619:160..256
@M38620:11..307
@M38621:11..179
@M38622:11..176
@M38623:11..250
ttccggagctgtatcatgcagctgtttggaaagaaggtggaggatgcatc
agaggtttccggctctaccacagaagtttctacagcctcgtaa
```

AFTER SECOND PASS:

```
>ASYPIGG6:CDS1
atggccgcacacgagcctgtgttcgccgcccggcgccacaatgaagacac
cacaagggagtctgcatttgtctacacaaatgctaataatacaagag
atccttttgaaggaccaaactatcacattgcccctcgatgggtctacaac
gtatcatccttatggatgatctttgttgtcattgcatcagtcttcactaa
tggtttggtaattgtagcaacagcaaagttcaagaagctgcgacaccctc
taaactggattctggtaaacctggctatagccgatctcggggagacagtt
cttgccagcacaatcagtgtcatcaaccagatcttcggctacttcatcct
tggacacccaatgtgcgtttttgaggggtggacggtgtctgtctgtg
gtatcacagctctgtggtctctgactataatctcctgggagcgctgggtg
gttgtgtgcaagccatttggaaatgttaaattcgatggcaaatgggcagc
aggtggcatcatcttctcctgggtttgggccatcatctggtgcacccctc
caatctttggctggagcag
gtactggccccatggtctgaagacatcctgtggccctgatgtgttcagtg
gcagtgaggatccaggagtggcctcctacatgatcaccctaatgcttacc
tgctgtattcttcctctgtccatcattatcatttgctacatttttgtctg
gagtgccatccaccag
gtcgcccagcagcagaaagactcagagtccactcagaaggcagagaagga
agtgtccaggatggtggtagtgatgatccttgcctttattgtgtgctggg
gaccatatgcctcctttgccaccttctctgcagtgaacccaggttatgcc
tggcacccactggcagccgctatgcccgcttacttcgccaagagtgccac
catctacaatcccatcatttacgtcttcatgaaccgccag
ttccggagctgtatcatgcagctgtttggaaagaaggtggaggatgcatc
agaggtttccggctctaccacagaagtttctacagcctcgtaa
```

Figure 5. Evaluation of segmented features.

To create this coding sequence, sequences from five different entries must be joined with bases 11..103 from ASYPIGG6. FEATURES accomplishes this task in a two step iterative process. First it evaluates as much of the expression as possible within the entry itself, such that after the first pass through the data the .out file would contain unresolved

```
misc_feature    X13383:1..734
                /label=drr49a_cDNA
unsure          "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
                 /label=polyA_tail
                /note="About 50bp but precise length unknown"
misc_feature    "tgcagg"
                /evidence=EXPERIMENTAL
                /label=Pst_to_Sal
                /note="pUC9 polylinker, from T-tailed PstI site to
                SalI"
misc_feature    X52331:join(735..2958,1..719)
                /label=KSm13_Sal_H3
                /note="Bluescript KSm13+/SalI/HindIII cut"
misc_feature    "agcttggctgcacccccccccccccccccc"
                /label=polyC_tail
                /note="pUC9 polylinker from HindIII to C-tailed
                PstI site"
contig          join(drr49a_cDNA,polyA_tail,Pst_to_Sal,KSm13_Sal_H3,
                polyC_tail)
                /organism="Escherichia coli"
                /plasmid="pI49KS"
                /label=pI49KS
```

Figure 6. Features Table for a synthetic sequence. A cDNA was originally cloned into the PstI site of pUC9 using poly-dC and poly-dT tailing. The insert was excised using SalI and HindIII and recloned into the corresponding sites in Bluescript KSm13+. In this entry, the first five features encode the components of the plasmid, while the 'contig' feature joins them to create the final construct. The 'contig' feature key is an extension of the FT definition understood by GETOB.

```
contig        join(J01619:1..13063,poly("n",7140),
              J03939:1..1363,poly("n",14380),
              X02306:complement(1..1622),poly("n",14710),
              J04423:1..5793,poly("n",22500),
              X03722:1..2400,poly("n",123750),
              one-of(X05017:complement(1..1854),X05017:1..1854))
              /label=Eco_contig8
              /map=763.4-950.6kb
contig        join(V00352:1..2412,poly("n",28800),M15273:1..3409)
              /label=Eco_contig9
              /map=972.9-1001.7kb
contig        join(X02826:1..1357,poly("n",13540),
              J01654:complement(1..2270))
              /label=Eco_contig10
              /map=1016.5-1031.4kb
chromosome    join(Eco_contig8,poly("n",22300),
              Eco_contig9,poly("n",14800),
              Eco_contig10)
              /label=Ecoli_chromosome
              /partial
```

Figure 7. FT representation of chromosomes and contigs. The feature keys 'contig' and 'chromosome', and the operator 'poly' are extensions to the FT definition that are understood by GETOB. **poly(***expression*,*x***)** is evaluated to mean that the expression is repeated x times. The part of the *E. coli* chromosome represented in this example runs from 763.4 - 1031.4kb, using data from [14].

expressions and sequence from ASYPIGG6. Next, accession numbers from unresolved expressions (denoted by '@') are used to retrieve entries from the database. In the second pass, the remaining expressions in the .out file are evaluated and replaced by sequence, with line breaks between each sequence region inserted for clarity. Since evaluation of an expression could potentially return another expression, FEATURES can iterate the process as many times as necessary to until no unresolved expressions remain. Any expressions that can not be resolved are 'commented out' of the file, meaning that a semicolon would be placed in front of the '@' to prevent FEATURES from looping infinitely, and a warning message would appear in the message file.

APPLICATIONS

Customized entries

It is possible to describe any sequence as a FT expression. FT expressions allow the construction of 'metaentries', that is, entries containing little or no sequence, with references to other entries, thereby specifying larger features. A feature table documenting the construction of a recombinant plasmid, as well as specifying FT expressions to generate its sequence, is shown in Figure 6.

FT expressions could be included in publications describing synthetic sequences such as cloning vectors or other recombinant constructs, providing an unambiguous definition of which fragments are present in the vector, as well as an easy formula for reconstructing the vector's sequence. Laboratories which make large numbers of synthetic constructs could maintain a database of metaentries documenting constructs in a precise and universally understood fashion. When plasmids are sent to other labs, an accompanying metaentry file would make it possible for the recipient to precisely reconstruct the plasmid sequence. When modifications are made to the plasmid, a new metaentry can easily be generated from the original to document those modifications.

The example in Figure 6 also illustrates the need for new feature keys to be added to the FT language definition to accommodate synthetic sequences. In place of 'contig', for example, the feature key 'plasmid' might be appropriate.

Working with very large sequences

The representation of large objects, such as chromosomes, will require increasingly complex descriptions, and often a merging of many pre-existing entries to form a larger entry. There are problems, however, with merging entries: a) absolute coordinates of virtually all features will have to be changed to reflect being part of a larger sequence b) direct merger of smaller entries would result in very long sequences that may be inconvenient to work with. Figure 7 is an example of how a portion of the *E. coli* chromosome might be represented in a metaentry. First, segments of the chromosome known to be contiguous are annotated as 'contig's, with spacing given by the poly operator. The chromosome, in turn, can be built by joining contigs. In this example, long stretches of n's serve as place holders for unsequenced regions. It is worth noting that the use of labels in these expressions means that changes in one feature do not necessarily require changes in higher-level features that use that feature. For example, if additional sequence entries were added to the feature labelled 'contig8', the chromosome feature would not need to be changed.

As larger and larger entries are assembled from many smaller entries, it becomes less convenient to work with individual sequence entries. That is, using conventional sequence tools, it is much easier to find and extract a tRNA sequence when it is the only one in the entry, than it is to find one of many tRNAs in a feature table, and to extract that particular sequence. As illustrated in the Figure 2, any tRNA in the *E. virginiana* plastid genome could easily be extracted using a simple feature expression.

Restriction maps

Because there is no standard way of representing restriction maps, the ability to communicate large maps such as the *E. coli* genome [15] in a computer-readable fashion is limited. As described above, the **poly** operator makes it possible to specify spacing between regions of known sequence, suggesting that feature expressions could be used to represent restriction maps. Figure 8 demonstrates the encoding of a restriction map into the FT language. Since this expression

```
contig      join("gaattc",poly("n",1200),"aagctt",
             poly("n",3430),"gaattc")
            /label=geneX
```

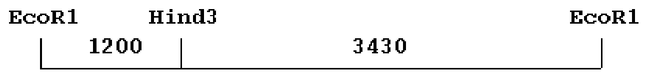


Figure 8. FT representation of restriction maps.

FEATURE TABLE:

```
mutation        replace(37308,"c")
                /label=cIam504
                /note="c in cIam504, g in wild-type"
mutation        replace(38302,"a")
                /label=cin-1
                /note="a in cin-1 , g in wild-type"
mutation        replace(45352,"a")
                /label=sam7
                /note="a in sam7 , g in wild-type"
mutation        replace(35940,"a")
                /label=rex209
                /note="a in rex209 , g in wild-type"
```

EXPRESSION:

```
J02459:group(1..48502,cIam504,cin-1,sam7,rex209)
```

Figure 9. FT representation of mutations in LAMBDA (J02459). For illustrative purposes, 'label' fields have been added, and errors in the original annotation corrected.

directly evaluates to a DNA sequence, any restriction map program that can read a sequence can make use of the information encoded in this fashion. Representation as an expression takes up minimal space, compared to bit-mapped graphics or the resultant sequence. Furthermore, as additional sequence or restriction site data becomes available, the expression can be modified to create an improved model of the sequence.

Mutants

The FT language provides the 'replace' operator as a means of representing changes in sequences, such as allelic variations or corrections to old versions of sequences. While replace is not extensively used in the database, it potentially simplifies the representation of sequence variants in a simple fashion. A sample metaentry representing mutations in the Lambda phage genome is shown in Figure 9. Each mutation, denoted by a unique label field, is annotated with an expression which, upon evaluation, changes the reference sequence to the mutant sequence. Below the metaentry is a FT expression that would generate a Lambda mutant of the genotype *cI, cin sam rex*. Entries constructed in this manner make it trivial to generate a model of essentially any genotype, based on the reference sequence. Only one reference sequence would need to be stored in the database if mutations were annotated in this fashion.

DISCUSSION

The FT language can serve as a platform-independent language for nucleic acid sequence manipulation. It is at the same time human-readable and machine-parsable. Since FT expressions refer only to database entries, they can be evaluated without the use of filenames or other system-dependent conventions. Even the format in which the database is stored, or its physical location, whether local or across a network, is irrelevant to the evaluation of FT expressions. If written with strict adherence to the FT definition, any program that evaluates FT expressions should return the same sequence. As a case in point, GenBank is now maintained at the National Center for Biotechnology Information (NCBI) as part of the GenInfo Backbone, using the Abstract Syntax Notation (ASN.1) [16]. GenBank flatfiles represent a report containing only a subset of information present in the GenInfo backbone. However, because all Features Table information is present in GenInfo, it should be straightforward to write a program comparable to FEATURES that extracts features from GenInfo.

The XYLEM tools are versatile and easy to use. High-level programs such as FINDKEY, FETCH and FEATURES can be run as Unix commands with options specified on the command line. This facilitates their use in combination with other software tools, such as GDE, which provides a graphic user interface. Command line execution is particularly useful in Unix, where output of one command can be piped into another command. Thus,

```
features M60287:ORF2 | ribosome
```

will cause the feature labelled 'ORF2' in entry M60287 to be evaluated, and the resultant sequence translated by the RIBOSOME program. At the same time, these high level tools have menu-driven user interfaces that can run on any text-only terminal. Where necessary, the low-level tools called by these high-level tools can also be directly accessed as Unix commands.

XYLEM provides a comprehensive set of tools for managing large databases for general use, as well as creating datasets customized for answering specific biological questions. While designed to automate dataset creation, human intervention is possible at all phases of the process. Upon the initial retrieval of hits from a FINDKEY search, the namefile can be edited to eliminate undesired entries. After the initial dataset has been created by FETCH, further operations can create additional datasets. For example, a study of nuclear-encoded plastid proteins might begin by using FINDKEY to identify all entries bearing the "transit_peptide" feature key. A subsequent search of the resultant dataset might eliminate mitochondrial entries while retaining plastid entries. Overlapping virtual datasets, consisting of LOCUS names or accession numbers could be created by grouping entries by species, or by function (eg. location in thylakoid membranes vs. lumen). FEATURES could be used to create separate files of transit peptides (transit_peptide) or mature peptides (mat_peptide). Inspection of the message file provides detailed information on each feature extracted, in the form of feature qualifier lines. Along with the expression itself, the qualifier lines can aid in the decision making process on use of the data, as in the case of pseudogenes or partial sequences that might need to be omitted from the dataset. At the same time, the message file has proven very useful in uncovering errors in database entries. Erroneous or otherwise undesired FT expressions can be deleted or corrected to allow generation of the final dataset. Thus implemented, the XYLEM tools provide both a high level of automation, while at the same time making the process of human intervention quick and painless.

The generality of the FT language makes it useful even for working with data that has not been stored in public databases. Customized entries can serve as a precise definition of a recombinant construct which can be used to automatically generate the actual sequence. FT expressions take up negligible space, as compared to the sequence itself, and can be modified more easily. Extensions to the FT language allowed by GETOB facilitate the construction of models of large genomic regions. Even where little or no actual sequence data is available, FT expressions can still be used to represent restriction maps. Their utility as a means of documenting mutations has also been demonstrated.

It has already been shown that the FT language still lacks some capabilities necessary for describing many types of molecules. One important shortcoming of the FT language is the lack of syntax for relative coordinates. For example, one might wish to retrieve all introns from a set of entries, accompanied by 25 bases of flanking sequence at both the 5' and 3' ends. In DELILA [5], the command might read

GET FROM INTRON BEGINNING - 25 TO INTRON END +25

While comparable FT expressions could be constructed using base ranges, FT syntax allows no general solution.

The FT language has additional inadequacies. For example, while the power of labels has been demonstrated in this paper, there is still no universal convention for creating labels that are unique and stable within the database. One possible solution would be for labels to be replaced by unique ASN.1 tags. In any case, protection of feature expressions from change demands that all features have labels, and features referring to other sequences use labels instead of base ranges. For this and other reasons, the FT language needs to undergo further evolution to realize its full potential.

I have demonstrated that the FT language, if used to its fullest by sequence annotators, can make the difference between data being merely human readable, to becoming computer-parsable. While the FT language may not be the last word in sequence annotation, it is important to recognize that even an imperfect standard, if used, opens the database for automated access in a way that a free text annotation can not. Additionally, coding of features into a computer-parsable form means that, should some better feature description language be incorporated into the databases in the future, existing expressions can be translated into the new language automatically.

ACKNOWLEDGEMENTS

This work owes a debt to the efforts of Thomas Schneider and Gary Stormo, who first articulated the unbreakable link between database definition and sequence query languages, as well as many of the fundamental ideas developed here. During the initial stages of this work the author was supported by a postdoctoral fellowship from the North Carolina Biotechnology Center, using a Unix 3b2 workstation donated by AT&T. Latter stages of the work were supported by the Natural Sciences and Engineering Research Council of Canada, Operating Grant OGP0105628, as well as the Bank of Nova Scotia and the University of Manitoba Faculty of Agriculture Endowment Fund. Thanks to Kathy Norman and Pat Macdonald for providing computer support services, and to Stuart Brown and Sandhya Tewari for their comments on the manuscript.

REFERENCES

1. Stormo, G.D., Schneider, T.D and Gold, L.M. (1982) Characterization of translational initiation sites in *E.coli*. *Nucl. Acids Res.* **10**, 2971-2996.
2. Jurka, J. and Milosavljevic, A. (1991) Reconstruction and analysis of human Alu genes. *J. Mol. Evol.* **32**, 105-121.
3. Burks, C., Cassidy, M., Cinkosky, M.J., Cumella, K.E., Gilna, P., Hayden, J.E-D., Keen, G.M., Kelley, T.A., Kelly, M., Kristofferson, D., and Ryals, J. (1991) GenBank. *Nucl. Acids Res.* **19** (Suppl), 2221-2225.
4. Schroeder, J.L. and Blattner, F.R. (1982) Formal description of a DNA oriented computer language. *Nucl. Acids Res.* **10**, 69-84.
5. Schneider, T.D., Stormo, G.D., Haemer, J.S. and Gold, L. (1982) A design for computer nucleic-acid sequence storage, retrieval and manipulation. *Nucl. Acids Res.* **10**, 3013-3024.
6. Read R.L., Davison, D., Chappelear, J.E. and Garavelli, J.S. (1992) GBPARSE: a parser for the GenBank flat-file format iwth the new feature table format. *CABIOS* **8**, 407-408.
7. The DDBJ/EMBL/GenBank Feature Table: Definition, Version 1.04, September 1, 1992.
8. Jensen, K. and Wirth, N. (1974) Pascal User Manual and Report. Springer Verlag.
9. Pearson, W.R. (1990) Rapid and Sensitive Sequence Comparison with FASTP and FASTA. *Meth. Enz.* **183**, 63-98.
10. Smith, S. (1993) Genetic Data Environment 2.2, unpublished.
11. Sidman, K.E., George, D.G., Barker, W.C. and Hunt, L.T. (1988) The protein identification resource (PIR). *Nucl. Acids Res.* **16**, 1869-1871.
12. Pfeiffer, F. and Gilbert, W.A. (1988) Vecbase, a cloning vector sequence data base. *Protein Sequences & Data Analysis* **1**, 269-280.
13. Cameron, G. (1988) The EMBL Data Library. *Nucl. Acids Res.* **16**, 1865-1867.
14. Rudd, K. E. (1990) Alignment of *Escherichia coli* K12 DNA sequences to a genomic restriction map. *Nucl. Acids Res.* **18**, 313-321.
15. Kohara, Y., Kiyotaka, A., and Isono, K. (1987) The physical map of the whole *E.coli* chromosome: Application of a new strategy for rapid analysis and sorting of a large genomic library. *Cell* **50**, 495-508.
16. Ostell, J. (1990) The GenInfo ASN.1 Syntax: Sequences. *Tech. Rep.* **1**, National Center for Biotechnology Information, National Library of Medicine, USA.