GENESIS, a knowledge-based genetic engineering simulation system for representation of genetic data and experiment planning

Peter Friedland¹, Larry Kedes², Douglas Brutlag³, Yumi Iwasaki¹ and Rene Bach¹

Departments of ¹Computer Science, ²Medicine, and ³Biochemistry, Stanford University, Stanford, CA 94305, and ²Veterans Administration Medical Center, Palo Alto, CA 94304, USA

Received 15 September 1981

Abstract

We have built a knowledge-based genetic engineering simulation system-- *GENESIS*-- capable of representing both domain-specific and general knowledge. Information is stored within a hierarchically-organized framework composed of structures called *units*. A series of sophisticated editors enables non-computer specialist molecular geneticists to construct a knowledge base through *direct* interaction with the computer. Three types of knowledge specific to the domain of molecular genetics, MAPS, SEQUENCES and RULES are discussed in detail with examples.

Introduction

Only within the last few years has it become clear that the domain of molecular biology has a widespread need for automated methods for experimental assistance. The advent of rapid DNA cloning and sequencing methods has had an explosive effect on the amount of data generated. This is most readily represented and analyzed by computer. Moreover, we have already reached a point where progress in the analysis of the information in DNA sequences is being limited by the combinatorics of the various types of analytical comparison methods available. The application of judicious rules for the detection of profitable directions of analysis and for pruning those which obviously lack merit will have an autocatalytic effect on this field in the immediate future.

Our research efforts have touched on four broad areas interfacing the data and methods of molecular genetics with the constructs of symbolic computation and artificial intelligence: representation, knowledge base development, planning, and immediate practical applications systems. Representation work focuses on the problem of acquiring, storing, and manipulating complex and varied information in the domain of molecular biology. Knowledge base development uses these representation methods to build encyclopedic collections of information for various areas of molecular biology. Planning research involves taking a scientist-provided description of an experimental goal, and, reasoning from the information in a knowledge base, constructing a chain of actions which should satisfy that goal. Applications systems take the form of smaller, stand-alone programs that accomplish relatively straightforward, but important, laboratory analyses. This paper deals with some of the ways we have approached the problem of representation and knowledge base building. Examples of planning and applications programs will appear elsewhere [1], [2].

Software systems, whether complex or relatively simple, should focus on making computer-naïve scientists capable of performing sophisticated computational tasks with a minimum of training. This requires a great deal of concentration on the human engineering aspect of computer/user interfaces so that the advanced performance features of modern programming are transparent to the user. The knowledge-based system described in this paper utilizes a complex internal method of representation to provide for its flexibility and power. However, the user only needs to understand a simple, self-documenting, English-like command language. Furthermore, the system is extensively interactive (help is available on-line at every point and the programs prompt the user for the next input required) and the programs and command structure are forgiving (typing errors and misspellings are corrected or queried by the program and mis-commands are gently and informatively explained).

Representation

Representation is considered the central problem in symbolic computation. The domain of molecular biology is particularly interesting because of the richness and variety of its information. This ranges from simple parametric values like pH and temperature to the linear representation of nucleic acid sequences as strings of the letters A, G, C, and T to complex maps of large molecules which indicate the locations of restriction cutting sites and complex functional regions (genetic loci). The information that must be represented also includes the procedural knowledge for manipulating other information (such as rules for checking the consistency and completeness of data), for selecting among alternative techniques in designing an experiment, and for describing overall strategies for reasoning in molecular biology.

Knowledge Base Construction

The testbed of molecular biology has led to the development of a general-purpose knowledge representation and acquisition system known as the Unit System [3], [4], [5]. The Unit System as applied to the molecular genetics has been designated as *GENESIS*, in keeping with its role as a *GENetic Engineering Simulation System*. *GENESIS* allows the convenient entry of any type of knowledge, whether domain general like integers and tables or domain specific like nucleic acid sequences and restriction maps, directly from the domain experts. It provides for both novice and expert users, and offers knowledge base builders wide latitude in structuring their individual representations. The system includes an English-like language (*Genglish*) for manipulating all of the information stored within knowledge bases [6]. This language allows and encourages molecular biologists to become their own "programmers" without having to worry about the underlying representation structures of their knowledge bases. For example, the phrase:

```
JOIN SV40 FROM FIRST ECOR1 SITE TO FIRST BAMH1 SITE  
TO PBR322 FROM 250 TO 3000 INTO NEWSEQUENCE
```

will perform the clearly indicated operation: a specific fragment of the DNA tumor virus SV40, whose

DNA sequence is stored in the knowledge base, will be appended (JOINED) to a subsegment of the popular plasmid cloning vector pBR322, whose sequence is also stored in the knowledge base. In this particular example the SV40 fragment was specified by denoting the names of restriction endonuclease cleavage sites while the pBR322 segment was delimited by specific base locations. In actuality, the DNA sequences themselves need not be stored or entered in the knowledge base, but only instructions as to where such sequences are stored in the computer system. The resulting "new" combined molecule will be stored in an information slot called NEWSEQUENCE which the user can examine and manipulate further. The terminology and flexible syntax of the instructions are written in *Genglish* designed to allow the geneticist to give commands in the argot of the research genre.

The Units knowledge bases provide a mechanism for managing the explosive growth of data and strategies in many areas of molecular biology without the necessity of building special purpose systems for each area. Also, the molecular geneticists themselves have been able to design and build their own systems, avoiding the time and reliability problem of a knowledge base passing through the filter of a computer scientist intermediary. The knowledge bases have served as "intelligent encyclopedias," as simulation systems, and as training vehicles.

A distinction, albeit a somewhat fuzzy one, should be made between a genetics knowledge-based system and a data-based system. The following is a list of dimensions which jointly and individually contribute both to the distinction and to the increased difficulty of allowing modification of a knowledge base by a computer-naïve geneticist compared to a data base:

- Complexity--the information stored within a knowledge base of a problem solving system such as *GENESIS* tends to be relatively complex in that more abstruse concepts are defined than within data bases. For example, computers aside, it is much more difficult for humans to rigorously define rules for gene splicing than to provide a table of airline flight schedules.
- Symbolic vs. numeric information--knowledge bases contain a large amount of symbolic information, data bases tend to use symbols mostly as labels.
- Variety--knowledge bases may require a great variety of different information types.
- Interrelationships among pieces of information--knowledge bases tend to require more implicit pointers among entries to provide the semantics, or network, of a problem-solving system.
- Predefined vs. free-format--data bases tend to fall into a tabular format, where large data structures can be expanded or compressed, but their basic structure cannot be altered. Knowledge-based systems often require a much more involved structural alteration when new concepts are described to the system.
- Amount of "meta"-knowledge included--data bases do not generally encompass knowledge about how to use other knowledge, whereas knowledge bases often do in the form of strategies, rules, or other "higher-order" knowledge.

The Unit System For Knowledge Representation

The Unit System on which *GENESIS* is based has provided a convenient representation system for the description of declarative information. Knowledge is organized as a hierarchical collection of frames or "units" with each unit containing many "slots", each holding an individual piece of knowledge. The Unit System has many user-oriented features that smooth the knowledge acquisition process for declarative knowledge. The capabilities of the Unit System also allow for the direct description by the experts of procedural knowledge. *GENESIS* has evolved considerably over the five year period of its development. By far the largest force motivating changes in the system have been the desires of its many computer-naïve users. This motivation is indeed central; we view the major function of *GENESIS* as being a friendly interface between domain experts and internal, easily manipulable, computer representations.

A typical segment of the unit hierarchy of a user-generated knowledge-base is seen below. Each line represents an individual unit of knowledge, in this case a set of genes of interest. Each of the gene units inherits all of the *types* of attributes ("slots") that were established in the top unit, ANIMAL-GENES, but the knowledge entered in each attribute slot can be particular to each gene ("unit").

```
ANIMAL-GENES
  PROTEIN-CODING-GENES
    CONTRACTILE-PROTEIN-GENES
      ACTIN-GENES
        D.DISCOIDEUM-ACTIN-GENES
        D.MELANOGASTER-ACTIN-GENES
        HUMAN-ACTIN-GENES
        RAT-ACTIN-GENES
        YEAST-ACTIN-GENES
        SEA-URCHIN-ACTIN-GENES
        CHICKEN-ACTIN-GENES
      GLOBIN-GENES
        ALPHA-GLOBIN-GENES
          MOUSE-ALPHA-GLOBIN-GENE
          HUMAN-ALPHA-GLOBIN-GENE
            HUM-ALPHA-1
            HUM-ALPHA-2
          NON-ALPHA-GLOBIN-GENES
            CHICKEN-BETA-GLOBIN-GENE
            HUMAN-BETA-GLOBIN
            HUMAN-DELTA-GLOBIN-GENES
            HUMAN-GAMMA-GLOBIN-GENES
            HUMAN-ZETA-GLOBIN-GENES
            MOUSE-BETA-MAJOR-GLOBIN-GENE
            MOUSE-BETA-MINOR-GLOBIN-GENE
            RABBIT-BETA-GLOBIN-GENE
        HEAT-SHOCK-GENES
        HISTONE-GENES
          CHICKEN-HISTONE-GENE
            CH-01
          D.MELANOGASTER-HISTONE-GENES
          HUMAN-HISTONE-GENES
          MOUSE-HISTONE-GENES
          NEWT-HISTONE-GENES
```

SEA-URCHIN-HISTONE-GENES
 L.PICTUS-HISTONE-GENES
 LPA
 LPB
 LPC
 P.MILIARIS-HISTONE-GENES
 H19
 H22
 S.PURPURATUS-HISTONE-GENES
 PSP2-17
 SP-H1
 SP-H4
 SP-HISTONE-ORPHONS
 SPH30-1
 SPH2B0-1
 SP-LATE-HISTONES
 SPL-1
 SPL-2
 SPL-3
TETRAHYMENA-HISTONE-GENES
XENOPUS-LEVIS-HISTONE-GENES
YEAST-HISTONE-GENES
 S.CERVISICAE-HISTONE-GENES
INSULIN-GENES
 MOUSE-INSULIN-GENE
 RAT-INSULIN-GENES
INTERFERON-GENES
 HUMAN-CDNA-PRECURSOR
 HUMAN-LI-ALPHA
 HUMAN-LIA-CDNA
OVALBUMIN-GENE
OVOMUCOID-GENE
SILK-FIBROIN-GENES
SOMATOSTATIN-GENE
THFR-GENE
TUBULIN-GENES
 DROSOPHILA-TUBULIN-GENES
 X.LEVIS-TUBULIN-GENES

Representations Of Genetic Information

We discuss here three of the symbolic representations that have been specifically developed for molecular genetics: Nucleic Acid Sequences, Genetic Maps (including the ability to assign loci), and the Rule language (*Genglish*) that allows the user to manipulate the genetic datatypes, maps and sequences, in such a way as to simulate both the actions and the logic required to perform recombinant DNA experiments. Other kinds of data that can be represented in *GENESIS* (integers, tables, lists, text etc.) are of obvious importance in knowledge base building but are less specific to the domain of molecular genetics and will not be extensively discussed.

Representation of Sequences

Nucleic acid sequence data banks, whether augmented by literature search (exemplified by the libraries compiled by Margaret Dayhoff and her colleagues of the National Biomedical Research

Laboratory, Washington, D.C. and by Walter Goad and his collaborators at the Los Alamos Scientific Laboratories) or by direct electronic input from user laboratories, will require sophisticated computer management for storage, annotation, and retrieval. We provide a method for storage of sequence data directly within our knowledge bases, but also recognize that in most cases such internal storage would be wasteful and redundant. Thus, in *GENESIS* the Sequence Editor allows bibliographic information to take the place of an actual DNA sequence. The Sequence Editor is an interactive editor whose subcommands allow the user to enter the identifying information about a sequence or to enter sequences directly from the terminal. The editor also can replace the bibliographic pointer with the actual sequence but this usually proves unnecessary.

Here is an example of the format of sequence information stored in a unit of *GENESIS*. User responses are in underlined letters and our parenthetical commentary is in *italics*.

UE : ed PBR322 *(the user asks the Unit Editor to see some of the information that is stored about the plasmid pBR322)*

=EDIT *(the command ed was enough to be recognized)*

SlotEd : edit nuc
=NUCLEOTIDE-SEQUENCE

The user is now editing the knowledge in the unit pBR322. Since such knowledge is stored in "slots" he commands the Slot Editor to link him to the nucleotide sequence slot where he interacts with the specialized Sequence Editor.

Sequence Editor
Seqed: pr

NUCLEOTIDE-SEQUENCE:
Circular Sequence PBR322 4362 basepairs long
stored in file <CSD.SEQUENCES>PBR322.SEQ.1

(The DNA sequence of pBR322 is not stored within the knowledge base, but rather in a public file elsewhere on the computer system. Other subcommands of the Sequence Editor can replace the pointed to sequence with the actual sequence.)

Seqed : do
=DONE

Representation of Maps

The linear arrangement of genes lends itself easily to representation by linear maps. The Map Editor allows the user to enter interactively sites, regions, length and topology (linear or circular). Segments can be inserted and deleted and the map can be enlarged to any practical degree. For complex map structure with large numbers of overlapping sites (e.g. restriction sites) and regions (e.g. genetic loci on either strands), segments of the map can be examined by calling for the region

name alone or by specifying the map coordinates to be examined. The Map Editor allows the user the flexibility to enter alphanumeric symbols to denote regions or ends of regions on a map. In addition, the Map Editor can represent genetic regions that have been interrupted by deletion, inversion or transposition events and regions that are located on opposite strands. For an extensive and annotated example of use of the Map Editor the reader is referred to the lengthy presentation of gene manipulation simulation that appears in the latter part of this paper.

Representation of the 'Rules' of Molecular Genetics

Perhaps the most powerful aspect of *GENESIS* is its capacity for allowing the domain experts, in this case the molecular biologists, to define the rules that govern the relationships between the experimental data, the biochemical techniques, and the physico-chemical entities that are manipulated both theoretically and in the laboratory. This capacity is contained in both the general forms of representation (e.g., integers, tables, lists etc.) and in a simple English language developed specifically for the molecular geneticist, *Genglish*. By writing the "rules" of molecular genetics (e.g. IF STRUCTURE IS CIRCULAR THEN SET AGAROSE-MOBILITY TO GREATER-THAN-LINEAR ELSE SET AGAROSE-MOBILITY TO LINEAR and IF AGAROSE-MOBILITY EQUALS LINEAR THEN LIGATE FRAGMENT-1 TO END OF LEFT-ARM) the researcher can simulate experimental tasks, predict outcomes, and formulate decisive and efficient experimental strategies.

Genglish is not a formal programming language; its power lies in the fact that the rules written by the geneticist are parsed automatically into the powerful programming language INTERLISP. The user does not need to know INTERLISP, does not even need to know how INTERLISP internally represents such computational constructs as matrices, arrays, strings, etc. These representations are made transparent for the user who needs to know only that when he defined a variable such as FRAGMENT-1, it is a DNA sequence (or map) and not that it is internally represented as a string or an array. Rules like those written above utilize a large number of verb-forms (SET and LIGATE in the example cited above, JOIN in the earlier example) and a flexible syntax. While writing rules, the user has on-line access to an interactive help file that explains the function of the verb-forms and the limitations of syntax. Equally important, as the rules are parsed into INTERLISP, any errors are noted and simple explanations made to allow the user to rewrite the rule in a precise way.

Clearly, one way for domain experts to enter procedural knowledge would be for them to learn the underlying programming language of the knowledge base system, in this case INTERLISP. This would not be a very good solution: first, because of the large time investment required to teach domain experts to be proficient programmers, and second, because the underlying principle of making the knowledge transparent to any domain expert would be violated. Being able to access and understand the procedural strategies of the expert provider of such strategies is just as important to another expert user as being able to access and understand a list of any particular parametric property.

The solution we have chosen for allowing all types of procedural knowledge to be described is to provide the specially tailored language of *Genglish* as a simplified "programming language." This language allows domain experts to access and manipulate the various units and slots within their knowledge bases--the "variables" of their programming language. Much of the language is domain independent, as will be seen below, but its architecture allows parts to be specially tailored to the particular domain.

Overall Design of *Genglish*

The language for procedural knowledge is implemented as the Rule Language within the Unit System upon which *GENESIS* is based. Interaction with the Rule Language is by the same basic mechanisms as is used for the different forms of declarative knowledge. The user "talks" to a special editor which provides facilities for entry and correction of individual statements within the language. (It should be noted that the word "rule" in the context of the Rule Language is used to refer to a single statement in the language, rather than necessarily to a conventional condition-action production rule of standard programming languages e.g. if condition "A" then do "B").

Currently, rule statements may be one of the following forms:

- An **action** -- which has the result of modifying some existing information, creating some new information, interacting with the user, or controlling the flow of rules.
- A **conditional** (or boolean) expression -- a relation which governs when the action is to be applied.
- An **iterative** expression -- which specifies that the remainder of the rule statement is to be executed a number of times based upon some condition.

Rules are parsed into INTERLISP and interpreted in two discrete steps. At rule entry time, the syntax of the rule written by the user is checked and rules are then stored in an internal format of procedures and parameters, or variables, upon which the procedures will be applied. Since the rules at this point need not be interpreted, or "run", rules can be written about entities which don't yet exist in the knowledge base, and allows rules to be general to more than a single context. For example, in the rule statement above, the conditional statement IF AGAROSE-MOBILITY EQUALS LINEAR.... contains the parameter AGAROSE-MOBILITY that is a "slot" or attribute of each member of a large set of DNA structural units in *GENESIS*. At rule interpretation time, the precise DNA structural unit whose agarose-mobility slot will be examined in order to retrieve a value, depends upon the context of "where" the rule was directed to look. If the context had been set to the unit for the covalently closed circular plasmid pBR322 the value returned would be different than that if the context had been the unit for a linear DNA molecule such as a restriction endonuclease fragment. The rule interpretation system keeps track of units in the current context-- these include the unit from which a rule is run (where the rule slot is stored), units implicitly mentioned in the rule statements (while iterating over a group of units, for example), and units which are explicitly added to the context by the user.

Actions in Genglish

Actions may be classed either as domain-independent or specific to the domain of molecular biology. A few of the latter class currently implemented follow: (for each action, a definition and an example or two will be given)

CLOSE makes a linear map/sequence circular.

CLOSE MAP1 INTO MAP2
CLOSE SEQUENCE1 INTO SLOT CIRCULAR-SEQUENCE

DELETE allows a part of a map or sequence to be deleted.

The part to be deleted can either be a region (in case of a map) or be a fragment between 2 integer delimiters. The word "COMPLEMENT (OF)" or "REVERSE (OF)" can be used to indicate inverse-complement or to reverse the fragment to be excised. 5' and 3' can be used to specify the first and the last position on a map/sequence. These syntactical freedoms are applicable to all the verbs that operate on maps and sequences.

DELETE REGION R1-REGION IN MAP1 INTO MAP2
DELETE SEQ1 FROM M TO 3' INTO SEQ2

ENTERREGION adds new regions to a MAP.

ENTERREGION LAC FROM 1200 TO 3500 IN GENE-MAP
ENTERREGION BIG-T FROM LEFTEND TO RIGHTEND IN CODING-MAP ON COMPLEMENT
ENTERREGION CODING-REGION FROM 5' TO 300 IN MAP1 BROKEN BOTH ENDS

ENTERSITE adds sites to a MAP.

ENTERSITE ECORI AT 1200 IN RESTRICTION-MAP
ENTERSITE PROTEIN SITES AT LOCATIONS IN PROTEIN-MAP

EXCISE allows a part of a map or sequence to be excised and put into another map or sequence slot.

EXCISE REGION R2-REGION IN T4-MAP INTO MAP2
EXCISE MAP1 R1-REGION MAP2
EXCISE FROM L TO M IN SEQ1 INTO SEQ2
EXCISE COMPLEMENT OF L TO M IN SEQ1 INTO SEQ2

INSERT allows a (part of) a map or a sequence to be inserted into another map or sequence at a specified position. This verb works for both linear and circular maps and sequences.

INSERT REVERSE OF REGION-1 IN MAP1 INTO MAP2 AT 30 INTO MAP3
INSERT SEQ1 FROM 10 TO 20 INTO SEQ2 AT 30 INTO SEQ3
INSERT MAP1 MAP2 50 MAP3
INSERT FROM 5' TO 2450 IN MAP1 INTO MAP2 AT 50 INTO MAP3

JOIN concatenates (parts of) two maps or sequences. This verb only works on linear maps and sequences.

JOIN LAC-OPERON FROM K TO 3' WITH REGION INSULIN-CDNA IN MAP2 INTO MAP3
JOIN SEQ1 WITH SEQ3 INTO SEQ3

JOIN COMPLEMENT REGION-1 MAP1 WITH MAP2 COMPLEMENT 10
20 INTO MAP3

OPEN cuts a circular map/sequence at a specified site to make it linear.

OPEN MAP1 AT CUTTING-SITE INTO SLOT-L
OPEN SEQ1 INTO SEQ2

SET the basic verb for setting and changing the values of slots. Slots can be set to constants, to the values of other slots, or to the values of constants and other slots modified by various unary or binary operators like PLUS, DIVIDED, INTERSECTION, or GET-ELEMENT.

SET ENZYME TO SAL1.
SET TWIST TO LINKING-NUMBER MINUS WRITHE
SET RESTRICTION-SITE TO GET-ELEMENT SITENUMBER OF SITE-
LIST.

Some operations done with the SET verb are specific to the molecular genetics datatypes.

TOPOLOGY-OF gets the topology of a sequence or a map as either CIRCULAR or LINEAR

SITES-OF gets a list of strings containing the names of all the sites in a map

SITE-LOCATIONS-OF

gets a list of integers containing all the locations of the sites in a map

REGIONS-OF gets a list of strings containing all of the names of regions in a map

REGION-LOCATIONS-OF

gets a list of integers containing the starting and ending location of each region in
a map

REGION-TABLE gets a table containing full information about all of the regions in a map.

REGION-MAP enters the regions from a table as defined above into a map.

SET MYTABLE TO REGION-TABLE OF MYMAP
SET NEWMAP TO REGION-MAP OF GENETABLE NEW

SET-LENGTH sets the length of a map.

SET-LENGTH OF RESTRICTION-MAP TO 3456
SET-LENGTH OF NEW-MAP TO OLD-LENGTH

SET-TOPOLOGY is used to set the topology of maps to either circular or linear.

SET-TOPOLOGY OF RESTRICTION-MAP TO CIRCULAR
SET-TOPOLOGY OF SITE-MAP TO NEW-TOPOLOGY

SPLICE allows a part of a map or sequence to be spliced.

SPLICE R1-REGION IN MAP1 INTO MAP2
SPLICE SEQ1 FROM 30 TO 3' INTO SEQ2

TRANSCRIBE allows parts of sequences to be transcribed to other sequence slots. T's are automatically converted into U's.

TRANSCRIBE COMPLEMENT OF PHIX FROM 1235 TO 1450 INTO
NEWSEQ
TRANSCRIBE PBR322 FROM GENESTOP TO GENESTART INTO
CODING-SEQUENCE
TRANSCRIBE SEQ1 FROM 150 TO 3' INTO SEQ2

TRANSFER-REGIONS

copies regions from one map to another.

TRANSFER-REGIONS FROM 5' TO 1000 IN MAP1 TO 1000 3' IN MAP2
 TRANSFER-REGIONS FROM 100 TO 200 IN MAP1 TO MAP2
 TRANSFER-REGIONS FROM MAP1 TO MAP2

TRANSFER-SITES copies sites from one map to another.

TRANSFER-SITES FROM 5' TO 1000 IN MAP1 INTO 1000 3' in MAP2
 TRANSFER-SITES FROM 100 TO 200 IN MAP1 TO MAP2
 TRANSFER-SITES FROM MAP1 TO MAP2

WRITE-SEQUENCE

allows one to write nucleic acid sequences into files in the standard MOLGEN format.

WRITE-SEQUENCE NAMED LAMBDA FROM NUCLEOTIDE-
 SEQUENCE INTO FILE NEWPHAGE

Boolean Expressions

Conditional (or Boolean) expressions provide a mechanism for testing relationships between slots. They contain one or more relation, each with two values to be compared, at least one of which must be the name of a slot. Conditional expressions are of the form:

IF relations THEN actions [ELSE actions].

Simple binary relations--equals, greater than, less than, list inclusion and the like--are provided as well as a special relation for finding patterns in the MAP and SEQUENCE datatypes. This conditional relationship, CONTAINS (as in IF MYSEQUENCE CONTAINS ANY TATA-BOX SITES THEN...), allows powerful searching of sequences, maps, strings, and even text. CONTAINS finds strings or lists of strings within a sequence, map, string, or text and also retrieves the locations of where the matches are found. The reserved words LOCATION or LOCATIONS will hold the integral locations of the sites found, stored as a list of integers that can be stored and searched like any other list. The words LOCATION1 and LOCATION2 will hold as integers the first and second location found. CONTAINS also allows specification of the number of sites that must be found for the relation to be true (at the moment, NO means no sites can be found, ONE, 1, or UNIQUE means exactly one site, otherwise any number one or greater is allowed). It also allows reading frames to be specified (FRAME1, FRAME2, or FRAME3) if matches are only to occur starting at those sites.

Some examples:

```
IF LENGTH-OF DIRECT-REPEAT IS GREATER THAN 5 THEN ...
IF OPEN-FRAME CONTAINS ATG THEN...
IF RESTRICTION-MAP CONTAINS INTRONS THEN ...
IF MYSEQUENCE CONTAINS CODONS IN FRAME2 THEN ...
IF MYTEXT CONTAINS ONE KEYWORD THEN ...
IF RESTRICTION-MAP IS FILLED THEN...
IF 3SPICE-SITE IS NOT EQUAL TO 4SPICE-SITE THEN...
IF CUTTING-LIST INCLUDES ENZYME THEN...
```

Iterative Statements

Iterative statements allow users to iterate over groups of units, rows of a table, or elements of a list. If the iteration is to be over a hierarchical group of units, then the parent unit of the desired group is

specified. During iteration over a table, individual entries in the table may be retrieved or altered by reference to the column names. During iteration over a list, the words ELEMENT, NEXT-ELEMENT, and PREVIOUS-ELEMENT allow the obvious reference. For example:

```
FOR ALL DNA-STRUCTURES
FOR ALL PHAGES WITH LENGTH > 500 AND STRANDEDNESS EQUAL DOUBLE
FOR EACH ROW IN RESTRICTION-ENZYMES TABLE
FOR EACH ELEMENT OF CUTTERS LIST
```

Facilities for Making Rule-Writing Easier

As the power of the Rule Language has increased, so too have the facilities for "programming" in it. Users can make free use of common synonyms and "skip" words to make the rules more readable. Rules can be entered directly by means of a rule editor which allows insertion, deletion, and replacement of rule statements. Users can also access their favorite text editor to write and modify rule slots. As was previously stated, variables ("slots") specified in rules are not referred to until rule interpretation time, so that the slots upon which a given set of rules operates do not have to exist when the rules are written.

An Example of Genetic Manipulation Simulation with GENESIS

While it is important to be able easily to write rules such as those above, when it becomes apparent that some rules are generally useful and that only the variables change, the rules can be encapsulated and made to function interactively with the user as if they were, in fact, a program. In the example below, a set of rules was utilized that allows the manipulation of any DNA structures known to the knowledge base. Since locations on DNA structures are usually identified by referring to a specific nucleotide location, or a specific restriction-endonuclease enzyme cutting site, or a specific genetic function or topological feature, the rules were written so that the user could specify both the names of the molecules to be manipulated (in this case a transposition of an insertion sequence into a cloned interferon cDNA) and the manner in which the precise loci of the transposition events were to be specified. As before, user responses to the various editors are in underlined letters, while explanatory comments added by the authors to the typescript of the interaction are in *italics*.

apply recombination-rules

Recombination-rules has six current modes: EXCISING (E) a DNA fragment from a molecule, DELETING (D) a DNA fragment from a molecule JOINING (J) two molecules together, INSERTING (I) a DNA fragment into a recipient molecule, REPLACING (R) a region of one molecule with a region from another molecule and TRANSPOSING (T) a region of one molecule to another location on the same molecule.

```
TYPE ONE OF E, D, I, J, R, T, Q, OR HELP   t
ENTER DNA-STRUCTURE CONTAINING SEGMENT TO BE TRANSPOSED : pbr-ifa-is1
```

(The user had already created a circular DNA structure consisting of the the plasmid pBR322

containing a segment of human leukocyte interferon- α cDNA and a copy of the prokaryotic transposable insertion sequence IS1.)

DO YOU WISH TO SPECIFY THE REGION TO BE TRANSPOSED BY A REGION NAME, BY SPECIFIC SITES OR BY COORDINATES? (N, S, C, OR HELP)? h

If you want to specify a REGION of the DNA structure by the name assigned in its restriction-map then respond with the letter "N".

If you want to select the limits of the region by sites at its ends (such as restriction sites) then respond with the letter "S".

If you want to select the limits of the region by giving the coordinates of its ends then respond with the letter "C".

You will be prompted for the region name, the sites or the coordinates as appropriate.

(N, S, C, OR HELP)? n

ENTER NAME OF REGION (TYPE H FOR HELP) : h

THE REGIONS IN PBR-IFA-IS1 ARE

A179	B125	B125	C338
C338	F99	F152	E152
D82	IS1	A-FRAME	B-FRAME
C-FRAME	&D-FRAME	L-INVERTED-REPEAT	R-INVERTED-REPEAT
A87	A96	B112	C84
BLAC	D132	LEADER	CODON
SIGNAL-PEPTIDE	TRAILER	MRNA	

DO YOU WANT TO SEE (S) OR EDIT (E) THE MAP OF PBR-IFA-IS1 (S OR E OR RETURN)? e

Map Editor

MapEd: pr

(The user wants to view the map of the DNA structure. The IS1 segment had been inserted at position 974 of the circular map of pBR322 thus breaking the continuity of numerous pBR322 genetic loci as indicated. The interferon cDNA had been inserted at 4380 and also interrupts genetic regions of pBR322. The single line map below is highly compressed so that restriction site information is not legible and the symbols on the map overlap. However, the Map Editor allows examination of subsegments of the map or expansion of the map. The restriction site and genetic region coordinates are automatically updated with every manipulation of the map. To save space, the restriction-sites have been removed from most of the subsequent examples.)

Circular

Length 6004 base pairs

A179 region from 86 TO 622 indicated by |A179 and A179|
 B125 region from 696 TO 974 indicated by |B125 and > broken on the right
 B125 region from 1743 TO 1838 indicated by < and B125| broken on the left
 C338 region from 259 TO 974 indicated by |C338 and > broken on the right
 C338 region from 1743 TO 2040 indicated by < and C338| broken on the left
 F99 region from 492 TO 788 indicated by |F99 and F99|
 E152 region from 625 TO 974 indicated by |E152 and > broken on the right
 E152 region from 1743 TO 1848 indicated by < and E152| broken on the left

D82 region from 326 TO 571 indicated by |D82 and D82|
 IS1 region from 975 TO 1742 indicated by X
 A-FRAME region from 1001 TO 1210 indicated by |A and A|
 B-FRAME region from 1030 TO 1302 indicated by |B and B|
 C-FRAME region from 1224 TO 1724 indicated by |C and C|
 D-FRAME region from 1406 TO 1693 indicated by |D and D|
 on the complementary strand
 L-INVERTED-REPEAT region from 975 TO 992 indicated by +
 R-INVERTED-REPEAT region from 1725 TO 1742 indicated by +
 A87 region from 2315 TO 2575 indicated by |A87 and A87|
 A96 region from 2585 TO 2872 indicated by |A96 and A96|
 B112 region from 1947 TO 2282 indicated by |B112 and B112|
 C84 region from 3556 TO 3807 indicated by |C84 and C84|
 BLAC region from 4065 TO 4380 indicated by |BLAC and > broken on the right
 BLAC region from 5255 TO 5796 indicated by < and BLAC| broken on the left
 D132 region from 2285 TO 2680 indicated by |D132 and D132|
 LEADER region from 4381 TO 4438 indicated by | and |
 CODON region from 4439 TO 5008 indicated by [and]
 SIGNAL-PEPTIDE region from 4439 TO 4507 indicated by S and S
 TRAILER region from 5009 TO 5246 indicated by 3' and 3'
 MRNA region from 4381 TO 5246 indicated by | and |

114454930810273147628197422885812924143403333394243899695550825554755988
 25 616 1208 1614 2126 2580 2979 3390 3895 4318 4744 5199 5681
 17 597 1050 1439 1865 2293 2799 3343 3721 4178 4575 5085 5605
 ||D82|E15+←B|C|D-D←+1|B11|D132|6|-----|C84|-|BL|SS-----3'3|-BLAC|---|
 0 6004

ALU1 HAE3 PVU2 AOS1 MBO1 HINF1 HINF1 TAQ1 HAE3 ASU1 HINF1 HINF1 MBO2
 TAQ1 FNU4H1 HAE3 HINF1 AOS1 HPA2 HHA1 ECOR2 XHO2 HPA2 AVA2 HGIA1 HGIA1
 HPAAACY112HGIA1ACY1ORACY1HPH1AU2CAU202ECOR2MHPHPHPH1A102EHPH1HACY1HMB02

MapEd: pr 2 900 1800

(User wants to expand by 2X and examine a subregion.)

Linear

Length 900 base pairs

B125 region from 696 TO 974 indicated by |B125 and > broken on the right
 B125 region from 1743 TO 1838 indicated by < and B125| broken on the left
 C338 region from 259 TO 974 indicated by |C338 and > broken on the right
 C338 region from 1743 TO 2040 indicated by < and C338| broken on the left
 E152 region from 625 TO 974 indicated by |E152 and > broken on the right
 E152 region from 1743 TO 1848 indicated by < and E152| broken on the left
 IS1 region from 975 TO 1742 indicated by X
 A-FRAME region from 1001 TO 1210 indicated by |A and A|
 B-FRAME region from 1030 TO 1302 indicated by |B and B|
 C-FRAME region from 1224 TO 1724 indicated by |C and C|
 D-FRAME region from 1406 TO 1693 indicated by |D and D|
 on the complementary strand.
 L-INVERTED-REPEAT region from 975 TO 992 indicated by +
 R-INVERTED-REPEAT region from 1725 TO 1742 indicated by +

|-----+←+←+|Axxx|BxxxxxxxxxxxxxxxxxxxxxxxxxxxxA|xx|CxxxxxxxxxB|xxxxxxxx|
 900 1350

|xxxxxxxx|DxxD|xxxC←←←-----|
 1350 1800

MapEd: pr mrna

(User wants to examine just the interferon "mRNA" region of the map.)

Linear

Length 865 base pairs

LEADER region from 4381 TO 4438 indicated by | and |
 CODON region from 4439 TO 5008 indicated by [and]
 SIGNAL-PEPTIDE region from 4439 TO 4507 indicated by S and S
 TRAILER region from 5009 TO 5246 indicated by 3' and 3'
 MRNA region from 4381 TO 5246 indicated by | and |

||-S-----S-----3'-----3|
 4381 5246

MapEd: do

DO YOU WANT TO SEE (S) OR EDIT (E) THE MAP OF PBR-IFA-IS1
 (S OR E OR RETURN)? <CR>
 ENTER NAME OF REGION (TYPE H FOR HELP) : is1
 DO YOU WISH TO GIVE A SPECIFIC SITE OR A COORDINATE FOR INSERTION
 (S OR C) ? c
 ENTER COORDINATE : 4299
 DOES THE TRANSPOSITION EVENT INVOLVE CREATION OF
 TERMINAL REPEAT SEQUENCES AT THE INSERTION SITE? (Y OR N): y
 WHAT IS THE LENGTH OF THE TERMINAL REPEAT SEQUENCE? (INTEGER): 5
 DO YOU WANT TO DELETE THE TRANSPOSED ELEMENT
 FROM ITS ORIGINAL LOCATION? (Y OR N): y

(Precise excision is presumed excluding excision of the host repeat sequence. Imprecise excision
 could be simulated semi-automatically by using the DELETE option of RECOMBINANT-RULES)

THE INSERTED SEQUENCE IS IN SEQ
 THE INSERTED MAP IS IN MAPS
 SHOULD I ENTER THIS DATA INTO A NEW UNIT (Y OR N) ? n

(The new map was put in an attribute slot called MAPS and the new sequence stored in an attribute
 slot called SEQ. The user elected not to create a new unit of knowledge containing the newly created
 DNA structure.)

RuleEd: done
 SlotEd : ed maps
 =EDIT

Map Editor

MapEd: pr

Circular

Length 6009 base pairs

(Note below that the IS1 element has "transposed" into the 3' end of the interferon mRNA and has
 created new break points)

A179 region from 86 TO 622 indicated by |A179 and A179|
 B125 region from 696 TO 974 indicated by |B125 and > broken on the right
 C338 region from 259 TO 974 indicated by |C338 and > broken on the right
 F99 region from 492 TO 788 indicated by |F99 and F99|
 E152 region from 625 TO 974 indicated by |E152 and > broken on the right

D82 region from 326 TO 571 indicated by |D82 and D82|
 B125 region from 975 TO 1070 indicated by < and B125| broken on the left
 C338 region from 975 TO 1272 indicated by < and C338| broken on the left
 E152 region from 975 TO 1080 indicated by < and E152| broken on the left
 A87 region from 1547 TO 1807 indicated by |A87 and A87|
 A96 region from 1817 TO 2104 indicated by |A96 and A96|
 B112 region from 1179 TO 1514 indicated by |B112 and B112|
 C84 region from 2788 TO 3039 indicated by |C84 and C84|
 BLAC region from 3297 TO 3612 indicated by |BLAC and > broken on the right
 D132 region from 1517 TO 1912 indicated by |D132 and D132|
 LEADER region from 3613 TO 3670 indicated by | and |
 CODON region from 3671 TO 4240 indicated by [and]
 SIGNAL-PEPTIDE region from 3671 TO 3739 indicated by S and S
 TRAILER region from 4241 TO 4298 indicated by 3' and > broken on the right
 MRNA region from 3613 TO 4298 indicated by | and > broken on the right
 IS1 region from 4299 TO 5066 indicated by x
 A-FRAME region from 4325 TO 4534 indicated by |A and A|
 B-FRAME region from 4354 TO 4626 indicated by |B and B|
 C-FRAME region from 4548 TO 5048 indicated by |C and C|
 D-FRAME region from 4730 TO 5017 indicated by |D and D|
 on the complementary strand
 L-INVERTED-REPEAT region from 4299 TO 4316 indicated by +
 R-INVERTED-REPEAT region from 5049 TO 5066 indicated by +
 TRAILER region from 5067 TO 5071 indicated by < and > broken both ends
 MRNA region from 5067 TO 5071 indicated by < and > broken both ends
 BLAC region from 5260 TO 5801 indicated by < and BLAC| broken on the left
 TRAILER region from 5072 TO 5251 indicated by < and 3' broken on the left
 MRNA region from 5072 TO 5251 indicated by < and | broken on the left

|||D82|E15<E|B11|D132|96|-----|C84|-|BLA|SS-----<+B|C|D-D<>3|-BLAC|--|
 0

6009

Map ed:

pr 6 3600 6000

Linear

Length 2400 base pairs

BLAC region from 3297 TO 3612 indicated by |BLAC and > broken on the right
 LEADER region from 3613 TO 3670 indicated by | and |
 CODON region from 3671 TO 4240 indicated by [and]
 SIGNAL-PEPTIDE region from 3671 TO 3739 indicated by S and S
 TRAILER region from 4241 TO 4298 indicated by 3' and > broken on the right
 MRNA region from 3613 TO 4298 indicated by | and > broken on the right
 IS1 region from 4299 TO 5066 indicated by x
 A-FRAME region from 4325 TO 4534 indicated by |A and A|
 B-FRAME region from 4354 TO 4626 indicated by |B and B|
 C-FRAME region from 4548 TO 5048 indicated by |C and C|
 D-FRAME region from 4730 TO 5017 indicated by |D and D|
 on the complementary strand
 L-INVERTED-REPEAT region from 4299 TO 4316 indicated by +
 R-INVERTED-REPEAT region from 5049 TO 5066 indicated by +
 TRAILER region from 5067 TO 5071 indicated by < and > broken both ends
 MRNA region from 5067 TO 5071 indicated by < and > broken both ends
 BLAC region from 5260 TO 5801 indicated by < and BLAC| broken on the left
 TRAILER region from 5072 TO 5251 indicated by < and 3' broken on the left
 MRNA region from 5072 TO 5251 indicated by < and | broken on the left

Concluding Comments

The application of the methods of artificial intelligence to the current problems of molecular biology serves to make new types of experiments possible that were not conceivable before. Knowledge base techniques can save considerable experimental effort in specific cases. These techniques have led us to new ways of looking at old problems and have even suggested new experimental approaches. Several of our collaborators are beginning to look to their knowledge bases for day to day guidance.

In addition, our methods are well suited to several other functions in addition to our central goal of providing experiment design assistance. The knowledge bases can serve as rapid and accurate references for current methods in molecular biology. Since scientists can easily collaborate and share information in building the knowledge bases, they serve an important didactic function in addition to being infinitely flexible reference books. Even more importantly, we have found that the organization and presentation of data within the knowledge bases makes them especially suitable for reviewing and analyzing data "by hand" as it were. DNA sequence information and restriction nuclease maps are quite suited to linear display and the datatype representation we have built into our knowledge bases makes comprehension of the results very simple.

One of the areas on which we have chosen to concentrate our efforts is in the representation of knowledge necessary to simulate and plan experiments involved with the cloning of specific genetic regions. These methods allow us to study in detail the organization of the DNA sequences that make up any given gene. A knowledge base not only can describe the methods of molecular biology, but also can organize the methods into plans for the isolation, characterization, and alteration of genes.

Acknowledgments

This work is a part of the MOLGEN project, a joint research effort among the Departments of Computer Science, Medicine and Biochemistry at Stanford University. The research has been supported under NSF grant MCS80-16247. Computational resources have been provided by the

SUMEX-AIM National Biomedical Research Resource, NIH grant RR-00785-08, and by the Department of Computer Science.

We are extremely grateful to our many enthusiastic collaborators in both computer science and molecular biology. They provide the continual challenge to build systems that are scientifically significant to both domains. Special thanks to Professor Edward Feigenbaum of the Stanford Computer Science Department for creating the environment of the Heuristic Programming Project in which this work was performed and to Professor John Sninsky of the Albert Einstein College of Medicine for many hours of testing and advice.

MOLGEN is a registered trademark of the Board of Trustees of Stanford University. *GENESIS* is a registered trademark of IntelliGenetics Inc.

Address all correspondence to: Dr. L.H.Kedes, 151M, VA Hospital, Miranda Avenue, Palo Alto, CA 94304, USA

References

1. Bach, R., Friedland, P., Brutlag, D. L., and Kedes, L. H., "A Sequencing Strategy Advisory System", Submitted to Nucleic Acids Research
2. Brutlag, D. L., Clayton, J., Friedland, P., and Kedes, L. H., "SEQ: A Nucleotide Sequence Analysis and Recombination System", Submitted to Nucleic Acids Research
3. Friedland, P., "Knowledge-Based Experiment Design in Molecular Genetics," Computer Science Department Report CS-79-771, Stanford, October 1979.
4. Stefik, M. J., "An Examination of a Frame-Structured Representation System," Proceedings of the Sixth International Joint Conference on Artificial Intelligence, IJCAI, 1979, pp. 845-852.
5. Smith, R. G. and Friedland, P., "A User's Guide to the Unit System," Heuristic Programming Project Memo HPP-80-28, Stanford, December 1980.
6. Friedland, P., "Acquisition of Procedural Knowledge from Domain Experts," Proceedings of the Seventh International Joint Conference on Artificial Intelligence, IJCAI, 1981, pp. 856-861.