QUERY DRIVEN SIMULATION AS A TOOL FOR GENETIC ENGINEERS

John A. Miller†, Jonathan Arnold‡, Krys J. Kochut†,
A. Jamie Cuticchia‡ and Walter D. Potter†

† Department of Computer Science and
Artificial Intelligence Programs
‡ Department of Genetics
University of Georgia

ABSTRACT

Simulations/animations of genetic structures and functions, simulations of actual or conceived experiments, and animations of algorithms such as simulated annealing, which is used to reconstruct a chromosome from its clonable DNA fragments, will be useful to genetics researchers and students alike. In this paper, we discuss the design of an integrated simulation/object-oriented database system that can be used by genetic engineers to understand better and to visualize the objects of their study as well as their experimental procedures. Such a system can provide a solid foundation for Computer-Aided Genetic Engineering (CAGE).

1. INTRODUCTION

Storing genome mapping information on organisms is currently the major unsolved problem of the Human Genome Initiative. Relational databases are beginning to be used to store the vast amount of genetic information that is being collected [Cuti91b] and [Rudd90, Rudd91]. We are currently designing an object-oriented database to store such information. Changing to this paradigm produces many significant advantages: storage of complex objects and images, such as gel patterns, more importantly the integration (actually encapsulation) of methods with the data, and lastly, a ready made framework to plug new modeling and methodological modules into for exploration via simulation. In other words, the database stores the data and its behavior together as objects. Now in addition to knowing about a certain gene, the database can also know about its behavior (e.g., how it codes for a protein), the methods necessary to place it on a physical or genetic map [Pear88, Land87, Suls88], or the search methods to link it to existing databases, such as GenBank. The ability to store arbitrarily complex methods in the object-oriented database allows simulation/animation features to be tightly integrated with the database. This is the idea behind *query driven simulation* [Mill89, Mill90, Pott90, Mill91a, Mill91b, Mill91c, Mill91d, Koch91]. By simply submitting a query to a query driven simulation system, information about objects (which may be simulation generated) will be displayed. For example, the scientist can watch as a gene or DNA fragment is being integrated into a physical map.

2. BACKGROUND

We have developed a Contig Mapping and Analysis Package, CMAP [Cuti91b], which provides a foundation for computer-aided reverse genetics by organizing information about DNA fragments derived from an organism's genome into a physical map. The user can store a variety of information about a particular segment of DNA in this relational database. This information or collection of attributes on clones can be both descriptive, such as any genes contained in a particular DNA fragment, or experimental, such as hybridization profiles for comparison with other fragments. In addition, this relational database is coupled with one method for ordering the DNA fragments along the chromosome to form a physical map. The user interface is designed to minimize both the learning curve associated with database usage, while eliminating the possibility of entering data outside the ranges of acceptable attribute values, the database constraints, through error checking protocols. Queries are currently accomplished by the use of SQL (Structured Query Language), which is the de facto standard query language for relational databases (and is the basis for some object-oriented query languages). SQL gives the user the ability to formulate queries based on any combination of attributes contained within the database. In order to eliminate the need for novice users to learn SQL, an interface was designed allowing users to build queries by menu choices. The CMAP database is a first step toward the production of an object-oriented database that will allow researchers to store and retrieve information vital for the implementation of reverse genetics through the activation of queries and more complicated methods.

Physical mapping involves the generation of an ordered list of clones, a list which represents the linear arrangement of clones along a chromosome [Coul86, Olso86, Suls88, Link91], much as genetic loci are ordered with respect to their position on linkage maps of chromosomes. This process of going from a DNA fragment to its function, reverse genetics, relies on linking the data on a DNA fragment, such as its sequence, to other databases, such as GenBank and the genetic map. Making these linkages requires the activation of specific methods, such as FASTA [Pear88] for searches of GenBank to find related DNA sequences or proteins or MAPSEARCH [Rudd91] to place the restriction map of a clone in the context of the organism's restriction map [Koha87]. In this way a physical map and its associated databases become a powerful tool for carrying out systematically the approach of reverse genetics.

The CMAP software package is designed for physical mapping experiments. Clonal attributes, such as chromosome localization and associated known genes can be stored, as well as raw data useful in physical mapping. By using hybridization profiles of clones with oligonucleotide probes as a measure of overlap [Lehr90, Crai90], CMAP can also produce data files which can be used by programs to order the clones accurately [Mich87]. The degree to which two clones overlap determines the degree of similarity in their hybridization profiles. We define d_{ab} as the difference in hybridization profiles between clones a and b. Thus, if clones a and b differ in the hybridization results for 12 probes, then $d_{ab} = 12$. We define D as the sum of the linking d_{ab} 's across the entire order of clones. The mapping procedure involves the minimization of this value,

$$D = \sum_{i=1}^{n} d_{a(i)b(i)}$$

where $a(i) = b(i-1)$ for $i > 1$. This is essentially a Traveling Salesman Problem (TSP) where the size of the problem is quite large (the number of clones, n, is in the hundreds). Since TSP is an NP-Hard [Gare79] optimization problem, simulated annealing is used to solve this problem. (Simulated annealing is not guaranteed to give optimal solutions, but in practice tends to give highly satisfactory solutions.) Once an order of fragments has been reached, CMAP can update the linkage of clones with this method to create the linear arrangement within the database.

3. QUERY DRIVEN SIMULATION

Because of the difficulty of simulating large complex systems with traditional tools, new approaches have and are being developed. One group of interrelated approaches attempts simultaneously to make simulation modeling and analysis easier while at the same time providing enough power to handle more complex problems. This group includes the following important (overlapping) approaches: integrated simulation support environments, object-oriented simulation, and knowledge-based simulation. There are two aspects of simulation that can lead to overwhelming complexity. First, simulation modeling and analysis uses and generates a huge amount of information. The management of this information has traditionally been handled by limited ad hoc means. *Integrated simulation support environments* (e.g., TESS [Prit86, Stan87]) have been of considerable help in this area. Second, the design, implementation, verification, and validation of complex simulation models from scratch is a formidable task. The closely related approaches of *object-oriented simulation* (e.g., SIMULA/DEMOS [Birt87, Birt73], MODSIM II [CACI90] and *knowledge-based simulation* (e.g., KBS [Fox89, Redd86] ROSS [McAr85, Roth89], and DEVS-Scheme/SES/MB [Zeig87, Zhan89, Zeig90]) allow new models to be composed from existing models, thereby enhancing the process of model development. For example, there may be a model for how the DNA sequence is generated, as one moves along the chromosome, and this must be coupled with a second model of how the DNA sequence evolves over time.

Query driven simulation is a powerful approach to simulation modeling and analysis. It fits somewhere in the middle of the three approaches discussed above. The basic motivation or premise behind query driven simulation is quite simple. Simulationists or even naive users should see a system based upon query driven simulation as a sophisticated information system. They should be able to interact with it at whatever level of detail they desire. A system/environment based upon query driven simulation will be able to store information about or to generate information about the behavior of systems that users wish to study. For the most part, users interact with the system simply by formulating queries.

To provide all the sophisticated information management needs involved in simulation (e.g., parameters, statistics, histograms, graphs, animations, and models), a database management system with greater functionality than a relational database system is required. The heart of the system we are developing is an *object-oriented database system* called Active KDL† (Knowledge/Data Language) [Mill90, Mill91d, Pott91]. (Much of the current research and development [Kim89b] in the database field involves object-oriented database systems, as they provide more powerful constructs for structural and behavioral specification. Well-known systems include ODE [Agra89], ONTOS [Andr89], GemStone [Cope84, Maie86], Iris [Fish87], ORION [Kim89a, Kim90], UniSQL [Kim91], and POSTGRES‡ [Ston86]. Three biological examples include applications to the nematode [Rowl91], *E. coli* [Kazi90], and protein hydrophobic cores [Kemp90].)

Active KDL is designed to support the complex information needs of *engineering databases* and *expert databases*. The fact that Active KDL is built from solid theoretical foundations (i.e., object-oriented programming, functional programming, and hyper-semantic data modeling) allows Active KDL to meet the needs of demanding applications (e.g. simulation, model management, CAD/CAM, human genome initiative, and intelligent database applications such as a university data/knowledge base capable of advising students). In particular, Active KDL provides an integration of *model bases*, *knowledge bases*, and *databases*. Simulation inputs and outputs can be stored by Active KDL since it supports complex objects. More importantly, Active KDL also allows users to specify rules to capture heuristic knowledge and methods to specify complex behaviors or computations. Finally, Active KDL provides a simple mechanism for specifying concurrent execution, namely tasks embedded in active objects. These facilities provide a powerful mechanism for building simulation models out of pre-existing model components.

† Active KDL evolved from KDL which was designed by Potter and Kerschberg [Pott86, Pott87], as a language for a hyper-semantic data model known as KDM [Pott86, Pott88, Pott89]. Hyper-semantic data models enhance semantic data models with object-oriented features and elements from artificial intelligence.

‡ POSTGRES is also classified as an extended relational system.

3.1. The Active KDL Language

Object-types (or classes) form the foundation of Active KDL. They are the main building blocks of a database schema specification. An Active KDL database consists of a collection of objects. As prescribed by the classification abstraction, similar objects are grouped into object-types (or classes). Instances of an object-type (or class) are called objects. An example of an object-type is GB_Gene (standing for GenBank Gene). This object-type consists of the set of all gene objects which are currently stored in GenBank (see the schema design in Section 3.2). In general, an object (as a value) is an entity composed of other values whose types may be different. The syntax of an object-type definition (object-type ::=) is shown in Figure 1.

```
object-type ::=
    OBJECT_TYPE class-name HAS
        [ SUPERTYPES:   // Generalization
            class-name { , class-name };]
        [ SUBTYPES:     // Specialization
            class-name { , class-name }
                [ HIDING function-list ];]
        [ ATTRIBUTES:   // Aggregation
            { attribute-name: type-name
                [ WITH CONSTRAINT: constraint ];}]
        [ MEMBERS:      // Membership (Association)
            { member-name: [ SET OF | LIST OF ] class-name
                [ INVERSE OF member-name [(class-name)]]
                [ WITH CONSTRAINT: constraint ];}]
        [ CONSTRAINTS:  // Integrity Constraints
            { constraint; }]
        [ HEURISTICS:   // Derived Information
            { rule; }]
        [ METHODS:      // General Computations
            { method; }]
    END class-name;
```

Figure 1: Syntax of Object-Type Definition.

Each object-type specification begins with the definition of the name of a class in the OBJECT_TYPE clause.

Any object-type may be defined as a specialized (derived) form of one or more other object-types, called *supertypes*. For example, the GB_Gene object-type is a specialization of the Gene object-type which is itself a specialization of the DNA_Fragment object-type. Consequently, GB_Gene inherits from its supertype Gene which inherits from its supertype DNA_Fragment, thus forming an inheritance hierarchy. We distinguish single and multiple inheritance, in case one or more supertypes are provided, respectively. If multiple inheritance occurs, then the inheritance hierarchy is generalized into an inheritance lattice [Mill91d].

The functional approach is present in all aspects of Active KDL. An object-type (or class) may be viewed as an encapsulation of functions. In Active KDL there are four flavors of functions:

1. *Attributes* are stored functions (if the attribute refers to an independently existing object(s) it is called a *member* and identified as such). Attributes are descriptive properties of an object and provide an aggregation mechanism (an object may be composed of sub-objects), while memberwise attributes (i.e. members) indicate relationships to other independent objects and provide an association mechanism.

2. *Constraints* are Boolean functions. They are typically enforced when the database is updated.

3. *Heuristics* are functions or rules expressed using the query language. These can act as stored parameterized queries or as view definitions.

4. *Methods* are quasi-functions (may have limited side-effects) expressed using the database programming language. Since the database programming language is Turing complete, methods can be used to express any computation.

4. GENETIC ENGINEERING APPLICATIONS

Query Driven Simulation simplifies the geneticist's interface to simulation. If the geneticist desires to know how, say, some mapping method performs, he/she formulates a query. The system will respond by either (1) retrieving stored data, (2) deriving new data from the stored data, or (3) instantiating and executing simulation models of how genomes behave. This same idea can be applied to the physical mapping application, where the model or method that may need to be executed could be simulated annealing which is used to place a new clone into the physical map (see Example 1).

The only noticeable difference to the user is in response time. A very nice feature of database systems is that they provide users with a concise and easy-to-use interface to large amounts of data. This interface is in the form of a query language (e.g., SQL). The query language for Active KDL has the high level, easy to use character of SQL. However, Active KDL is strictly more powerful than SQL since it supports recursive queries (through heuristics) and general computations (through methods). Furthermore, its object-orientation allows not only data, but knowledge, via heuristics and methods, to be accessed. The heuristics and methods can be used to construct and manipulate the complex objects stored in the database (e.g., DNA sequences and physical maps [Fu91]). The power of the Active KDL database system allows for the development of sophisticated information intensive applications. Information about the structure, performance, and reliability of systems under consideration is captured by Active KDL. This information allows Active KDL to answer questions posed by users. The questions may be answered by simple data retrieval, complex query processing, querying requiring heuristic knowledge, or even model instantiation (e.g., simulation of protein coding or animation of the simulated annealing algorithm for reconstructing chromosomes).

Model instantiation [Mill90] occurs when Active KDL does not have sufficient data or knowledge to provide a satisfactory answer. Depending on the complexity of the query, model instantiation may be a simple or quite complex process. The process centers around the creation of sets of input parameter values which are obtained by schema and query analysis. These values are used to create model instances (via the instantiation process) which are then executed to produce results.

4.1. Example 1: Simulated Annealing

As an example of the use of model instantiation in the physical mapping problem domain, consider the following query which is to return the contig on which a clone is located and its relative position on this contig. This information is requested for all clones matching the index name where the row number is a wildcard.

```
FOR ALL c IN Clone WHERE Index (c) = "L22?11"
   APPLY Index (c), On_Contig (c), Order_Num (c)
END;
```

If all of these clones have already been placed on the physical map, then rapid retrieval will yield the desired information; otherwise model instantiation in the form of a simulated annealing method will be used to find a location on the physical map for each missing clone. In such a case, Active KDL automatically creates model instances that are executed to generate enough data to give a satisfactory answer to the user. To aid in understanding how chromosomes are reconstructed from clones, this simulated annealing process may be animated. Some of the more important object-types used in the integrated simulation/object-oriented database system are shown below; in particular, the reconstruct method is within the Physical_Map object-type.

```
OBJECT_TYPE DNA_Fragment HAS
    // Instances of this class are fragments of DNA
    // identified by an STS.  Their complete sequence
    // may or may not be known.
    MEMBERS:
        Identifying_STS: STS;
END DNA_Fragment;

OBJECT_TYPE Sequence HAS
    // Instances of this class are sequences of nucleotides.
    ATTRIBUTES:
        Nucleotides: LIST OF (A, C, G, N, R, T, Y);
            // Enumeration type allowing compaction
        Base_Count_A: INTEGER;      // Adenine
        Base_Count_C: INTEGER;      // Cytosine
        Base_Count_G: INTEGER;      // Guanine
        Base_Count_T: INTEGER;      // Thymine
            // N for anything, R for A or G, Y for C or T
    HEURISTICS:
        Base_Count (s: Sequence): INTEGER =
            Base_Count_A (s) + Base_Count_C (s) +
            Base_Count_G (s) + Base_Count_T (s);
        Ambiguity (s: Sequence): INTEGER =
            Size (Nucleotides (s)) - Base_Count (s);
    METHODS:
        FASTA (s: Sequence; t: Sequence): INTEGER;
            // Return accession where the t sub-sequence matches s
        Enter_Sequence (s: Sequence);
        Display_Sequence (s: Sequence);
            // Display using the letter (ascii) representation
END Sequence;

OBJECT_TYPE Clone HAS
    // Instances of this class are clonable DNA fragments.
    SUPERTYPES:
        DNA_Fragment;
    ATTRIBUTES:
        Index: STRING;              // Clone Identifier
        Location: Storage_Library;  // Location in Storage Library
        Active: BOOLEAN;            // Location contains viable clone?
        Date: STRING;               // Date Clone was isolated
        Clone_Comment: Comment;     // Misc. information about clone
        Repetitive: BOOLEAN;        // Clone has repetitive DNA?
        rRNA: BOOLEAN;              // rRNA is coded for in clone?
        Order_Num: INTEGER;         // Location on contig
        Clone_Size: INTEGER;        // Size of clone in kb
        Num_Probes: INTEGER;        // Number of oligo probes used
    // Hybridization profiles:  whether the clone hybridizes with ith
    // (i = 1..30) oligonucleotide probe (DNA sequence of 9 to 12 bases)
        Oligo: LIST OF BOOLEAN WITH CONSTRAINT
            Oligo_Size (c: Clone): BOOLEAN =
                Size (Oligo (c)) = Num_Probes (c);
    // Hybridization profiles:  whether the clone hybridizes with ith
    // (i = 1..8) chromosome, rated as High, Medium, or Low
        Chromo_Specific: LIST OF (H, M, L) WITH CONSTRAINT
            CS_Size (c: Clone): BOOLEAN =
                Size (Chromo_Specific (c)) =
                    Num_Of_Chromosomes (From_Organism (c);
    MEMBERS:
        Known_Genes: SET OF Gene; // Known genes contained within clone
        On_Contig: Contig;        // Contig to which clone is localized
        From_Organism: Organism;  // Source Organism
    CONSTRAINT:
        Index_Check (c: Clone): BOOLEAN =
            Index (c) = SubLibrary (Location (c)) + Plate (Location (c))
                + Row (Location (c)) + Column (Location (c));
    METHODS:
        Match_Gene (c: Clone): Gene;
            // Return gene (if there is one) whose STS matches clone's.
END Clone;

OBJECT_TYPE Contig HAS
    // Instances of this class are contiguous sections of chromosome
```

```
    // which result from piecing the clones back together.
    ATTRIBUTES:
        Contig_Name: STRING;
        Contig_Size: INTEGER;       // Size of contig in kb
        Gap_Size: INTEGER;          // Number of kb to next contig
    MEMBERS:
        Clones: LIST OF Clone INVERSE OF On_Contig;
        On_Chromosome: Chromosome;
    HEURISTICS:
        Current_Clone (k: Contig): Clone =
            Current (Clones (k));
        Next_Clone (k: Contig): Clone =
            Next (Clones (k));
        Prior_Clone (k: Contig): Clone =
            Prior (Clones (k));
    METHODS:
        Set_Current (k: Contig; c: Clone);
END Contig;

OBJECT_TYPE Chromosome HAS
    // Instances of this class are chromosomes.
    ATTRIBUTES:
        Chromosome_Num: INTEGER WITH CONSTRAINT
            Number_Check (c: Chromosome): BOOLEAN =
                Chromosome_Num (c) IN
                    {1 .. Num_Of_Chromosomes (From_Organism (c))};
    MEMBERS:
        From_Organism: Organism INVERSE OF Genome;
        Genes: LIST OF Gene
            INVERSE OF On_Chromosome (Gene);
        Contigs: LIST OF Contig
            INVERSE OF On_Chromosome (Contig);
        RFLPs: LIST OF RFLP;
    HEURISTICS:
        Clones (c: Chromosome): LIST OF Clone =
            Clones (Contigs (c));
END Chromosome;

OBJECT_TYPE Gene HAS
    // Instances of this class are genes.
    SUPERTYPES:
        DNA_Fragment;
    ATTRIBUTES:
        Name: STRING;
        Gap_Size: INTEGER;      // Number of kb to next gene
    MEMBERS:
        Effected_Trait: Trait;
        Generated_Proteins: SET OF Protein;
END Gene;

OBJECT_TYPE GB_Gene HAS
    // Instances of this class are genes having entries in GenBank.
    SUPERTYPES:
        Gene;
    ATTRIBUTES:
        Locus: STRING;
        Definition: STRING;
        Accession: STRING;
        Keywords: LIST OF STRING;
        GB_Gene_Comment: Comment;
        Features: LIST OF Feature;
        Origin: STRING;
        DNA_Sequence: Sequence;
    MEMBERS:
        From_Organism: Organism;
        Reference: Article;
        Sites: LIST OF Site;
    METHODS:
        Display_Sequence (g: GB_Gene);
END GB_Gene;

OBJECT_TYPE Map HAS
    MEMBERS:
        Of_Organism: Organism;
END Map;

OBJECT_TYPE Genetic_Map HAS
    SUPERTYPES:
        Map;
    METHODS:
        Display_Genome (g: Genetic_Map; w: INTEGER);
            // Display all the chromosomes in genome of the organism
            // in window w.  This provides a less detailed overview.
        Display_Chromo (g: Genetic_Map; n: INTEGER; w: INTEGER;
                genes: BOOLEAN; RFLPs: BOOLEAN; STSs: BOOLEAN);
            // Display interesting points along the n-th chromosome
            // where genes, RFLPs, and/or STSs are located.  To
            // achieve higher resolution which expands these points
            // of interest, a physical map will need to be displayed.
        Zoom (w: INTEGER; w2: INTEGER = w);
            // Enlarge the display of the genetic map which is
            // currently in window w.
        UnZoom (w: INTEGER; w2: INTEGER = w);
        Move   (w: INTEGER; distance: INTEGER);
        Switch (w: INTEGER; w2: INTEGER = w);
            // Switch to the corresponding physical map.
END Genetic_Map;

OBJECT_TYPE Physical_Map HAS
    SUPERTYPES:
        Map;
    METHODS:
        Reconstruct (p: Physical_Map; n: INTEGER; new: SET OF Clone);
            // Use simulated annealing to reconstruct chromosome n.
        Display_Chromo (p: Physical_Map; n: INTEGER; w: INTEGER);
            // Display the n-th chromosome as an ordered list of
            // overlapping clones.  Initially clones will be shown as
            // very short line segments that together to form contigs.
            // Group of contigs and gaps constitute the chromosome.
        Zoom (w: INTEGER; w2: INTEGER = w);
            // Enlarge display of the physical map which is currently
            // in window w.  As the display continues to zoom, details
            // of the clones will begin to appear (e.g., the STS,
            // contained genes, nucleotide sequence, etc.)
        UnZoom (w: INTEGER; w2: INTEGER = w);
        Move   (w: INTEGER; distance: INTEGER);
        Switch (w: INTEGER; w2: INTEGER = w);
            // Switch to the corresponding genetic map.
END Physical_Map;
```

4.2. Example 2: Simulated In Vitro Reconstruction of a Chromosome

The simulation and animation of planned experiments can be very useful to genetics researchers, their support staff, and their students. By simulating the experiment before it is carried out, the geneticist can refine and improve the experiment and possibly avoid pitfalls. After the experimental procedure has been refined, the laboratory staff and students can be trained using the simulation as a guide.

For example, an artificial chromosome can be generated on the computer [Arno88] by modeling a DNA sequence as a Markov chain. The database software could activate methods to simulate cloning of random DNA fragments of fixed size to create an artificial clone collection or library. In turn, the clones in the library can be hybridized to a battery of short synthetic DNA sequences of 9 to 12 bases. These synthetic oligonucleotides or probes hybridize to specific sequences in a clone, thereby assigning each clone a digital call number on the basis of hybridization or no hybridization with a given probe. If there is a high degree of overlap between two clones, they share similar call numbers. Clones can then be ordered with respect to their position along a chromosome by their call number, as in a real library. This process of clone/probe hybridization can also be simulated [Cuti91a].

Let us consider how this type of computer-aided experimental design might be applied to the problem of designing a clone hybridization experiment. Oligonucleotide probes need to be selected for the purpose of deciding to what degree various clones overlap. The experimenter also needs to decide how many probes are to be used to reconstruct the chromosome in vitro [Fu91], i.e., the length of the call numbers. Choosing the right number and right combination of probes, can have an important bearing on the completeness of the ultimate reconstruction. Being able to simulate the reconstruction experiment above would provide one means for an experimenter to decide how satisfactory a map is going to be.

Once the geneticist has chosen the set of probes, the staff and students can now be trained on how to conduct the actual experiment and on what kind of results to expect. They can see the true ordering of clones along the chromosome. They also can see the true contiguous blocks of overlapping clones along the chromosome or contigs, how long they are, and what percentage two random clones overlap. They can then implement a section of the experiment with a given number of specific probes, and display graphically the results of the experiment: 1) a sequence of filters with filled or open circles, depending on whether or not a probe hybridized to a specific probe; 2) statistics on how many clones hybridized with a given probe or pair of probes together with expectations; and 3) the probe hybridization sites highlighted on the true contig map. With a sense of what probes might be linked together in an inferred map, they can then activate the physical mapping method and see how they do. They would see how well they perform in identifying contigs and how long the contigs are. They would see how well they detect overlaps. Coupling the animation of the reconstruction experiment with the mapping method could provide useful insights into how the experiment will proceed. Other mapping experiments [Suls86] could be similarly implemented.

4.3. Example 3: Linking the Genetic Map to the Physical Map

A major avenue to inferring the function of a DNA fragment is to link it to the genetic map [Clut90, O'Bri90], where often the phenotype of a gene is known. To carry out this correspondence either the database must recursively establish that a given DNA fragment is part of a gene with known map position or part of a contig, in which at least one clone has a position on the genetic map. If a standard query cannot provide this information, then a method must be selected to estimate where a clone in the physical map falls on the genetic map. For example, if a clone is part of a contig for which hybridization data or chromosomal walk data is available, then a method can be activated to estimate the average overlap between clones and hence the average physical distance between a given clone and a clone in the same contig on the genetic map. Using either global regression of genetic map distances in kilobases between genetic markers or estimates of the physical size of a map unit [Brod91], the genetic map position can be estimated. Alternatively, local methods can be activated, in which two (possibly more) nearest (on the physical map) restriction site polymorphisms on contigs (or big DNA fragments of Yeast Artificial Chromosomes (YAC) [Burk87]) are used to triangulate on the genetic location of a given clone. When big fragments are cloned or a contig contains a given clone, it is likely [Meag88] under some circumstances that the big clone, like a YAC, or contig contains multiple RFLPs. The linkage should be graphical and dynamic. As the experimenter slides along the physical map, he should be able to slide along the genetic map as well.

If there is no information to make an estimate, at the least the program can take the contigs anchored on the genetic map and subtract out their projection onto the genetic map, to indicate possible locations for the given clone. A priori, the chance that the given clone falls in the remaining gaps should be proportional to the size of the gap, and so a probability density can be reported in the form of a histogram above the gaps.

This inference process can be simulated with the construction of two artificial chromosomes, as described in Example 2. RFLPs could be generated in two artificial chromosomes, which are assumed to have diverged from each other T generations ago according to a simple base substitution model, as described in [Meag88]. Crosses between the individuals homozygous for these two chromosomes can then be made to generate the RFLPs. The crosses and the RFLPs from random clones could be displayed on the computer for the experimenter, using various restriction enzymes. An RFLP map could be generated by activating MAPMAKER [Land87]. An experimenter could then try out the linking operation above on the artificial genetic and physical maps, seeing how many contigs become anchored on the genetic map, how well the inference process above works for estimating the genetic map position of a clone, and how concordant the orderings are of clones on the genetic and physical maps.

5. CONCLUSIONS

The storage of genome mapping information requires an enormous number of complex objects to be stored. To test and understand how these objects behave and interact, and to plan and explain experimental procedures, simulation/animation is an invaluable tool. Clearly, the integration of database and simulation tools is necessary to provide genetic engineers with capabilities needed for their most complex task. Query driven simulation provides such an integration in a tight, uniform and easy-to-use manner.

6. REFERENCES

[Agra89] R. Agrawal, and N.H. Gehani, "ODE (Object Database and Environment): The Language and the Data Model," *Proceedings of the SIGMOD International Conference on Management of Data* (June 1989).

[Andr89] Andrews, T., C. Harris, and K. Sinkel, "The ONTOS Object Database," *Ontologic Technical Report*, Burlington, MA (1989).

[Arno88] Arnold, J., Cuticchia, A.J., Newsome, D.A., Jennings, W.W., and Ivarie, R. (1988) "Mono-Through-Hexa Nucleotide Composition of the Sense Strain of Yeast DNA: A Markov Chain Analysis," *Nucleic Acids Research*, 16 7145-7158.

[Birt87] Birtwistle, G., *DEMOS: A System for Discrete Event Modelling on SIMULA*, Springer-Verlag, N.Y. (1987).

[Birt73] Birtwistle, G., O. Dahl, B. Myhaug, and K. Nygaard, *Simula Begin*, Studentlitertur and Auerbach Publishers (1973).

[Brod91] Brody, H., Griffith, J., Cuticchia, A.J., Arnold, J., and Timberlake, W.E. (1991) "Chromosome-Specific Recombinant DNA Libraries from the Fungus *Aspergillus nidulans*," *Nucleic Acids Research*, 19:11, 3105-3109.

[CACI90] CACI, *MODSIM II: The Language for Object-Oriented Simulation*, Los Angeles, CA (January 1990).

[Clut90] Clutterbuck, A.J. (1990) "*Aspergillus nidulans," Genetic Maps: Locus Maps of Complex Genomes*, Fifth Edition, O'Brien, S.J. (Ed.), Cold Spring Harbor Laboratory Press, Plainview, N.Y.

[Cope84] Copeland, G., and D. Maier, "Making Smalltalk a Database System," *Proceedings of the SIGMOD International Conference on Management of Data* (June 1984).

[Coul86] Coulson, A., Sulston J., Brenner S., and Karn, J. (1986) "Toward a Physical Map in the Genome of the Nematode *Caenorhabditis elegans*," *Proceedings of the National Academy of Sciences*, USA, 83, 7821-7825.

[Crai90] Craig, A.G., Nizetic, D. Hoheisel, J.D., Zehetner, G., Lehrach, H. (1990) "Ordering of Cosmid Clones Covering the Herpes Simplex Virus Type I (HSV-I) Genome: A Test Case for Fingerprinting by Hybridisation," *Nucleic Acids Research*, 18:9, 2652-2659.

[Cuti91a] Cuticchia, A.J., Arnold, J., and Timberlake, W.E. (1991) "Use of Simulated Annealing for Chromosome Reconstruction Experiments Based on Binary Scoring," *Genetics*, under review.

[Cuti91b] Cuticchia, A.J., Arnold, J., Brody, H, and Timberlake, W.E. (1991) "CMAP: Contig Mapping and Analysis Package: A Relational Database for Chromosome Reconstruction," *CABIOS*, under review.

[Fish87] Fishman, D., et al., "Iris: An Object-Oriented Database Management System," *ACM Transactions on Office Information Systems*, Vol. 5, No. 1 (January 1987).

[Fox89] Fox, M.S., N. Husain, M. McRoberts, and Y.V. Reddy, "Knowledge-Based Simulation: An Artificial Intelligence Approach to System Modeling and Automating the Simulation Life Cycle," in *Artificial Intelligence, Simulation and Modeling*, L.E. Widman, K.A. Loparo, and N.R. Nielsen (Eds.), Wiley Interscience, N.Y. (1989).

[Fu91] Fu, Y.X., Timberlake, W.E., Arnold J. (1991) "On the Design of Genome Mapping Experiments Using Short Synthetic Oligonucleotides," *Biometrics*, in revision.

[Gare79] Garey, M.R., and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco (1979).

[Kazi90] Kazic, T., Lusk, E., Olson, R., Overbeck, R., Tuecko, S. (1990) "Prototyping Databases in Prolog," in *The Practice of Prolog*, Sterling, L. (Ed.), MIT Series in Logic Programming, MIT Press, Cambridge, MA.

[Kemp90] Kemp, G.J.L., and Gray, P.M.D. (1990) "Finding Hydrophobic Microdomains Using an Object-Oriented Database," *CABIOS*, 6:4, 357-363.

[Kim91] Kim, W., "Object-Oriented Database Systems: Strengths and Weaknesses," *Journal of Object-Oriented Programming*, Special Issue on Databases, Vol. 4, No. 4 (July-August 1991).

[Kim90] Kim, W. *Introduction to Object-Oriented Databases*, The MIT Press, Cambridge, MA (1990).

[Kim89a] Kim, W., "A Model of Queries for Object-Oriented Databases," *Proceedings of the Fifteenth International Conference on Very Large Data Bases*, Amsterdam (1989).

[Kim89b] Kim, W., and F.H. Lochovsky (Eds.) *Object-Oriented Concepts, Databases, and Applications*, Addison-Wesley, Reading, MA (1989).

[Koch91] Kochut, K.J., J.A. Miller, and W.D. Potter, "Design of a CLOS Version of Active KDL: A Knowledge/Data Base System Capable of Query Driven Simulation," *Proceedings of the 1991 AI and Simulation Conference*, New Orleans, LA (April 1991).

[Koha87] Kohara, Y., Akliyama, K., 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.

[Land87] Lander, E.S., and Green, P. (1987) "Construction of Multilocus Genetic Linkage Maps in Humans," *Proceedings of the National Academy of Sciences*, USA, 84, 2363-2367.

[Lehr90] Lehrach, H., Drmanac, R., Hoheisel, J., Larin, Z., Lennon, G., Monaco, A.P., Nizetic, D., and Poustka, A. (1990) "Hybridization Fingerprinting in Genome Mapping and Sequencing," *Genome Analysis Volume I: Genetic and Physical Mapping*, Cold Spring Harbor Laboratory Press, Davies, K.E., and Tilghman, S.M. (Eds.), Plainview NY.

[Link91] Link A.J., and Olson, M.V. (1991) "Physical Map of *Saccharomyces cerevisiae* Genome at 110-Kilobase Resolution," *Genetics*, 127 681-698.

[Maie86] Maier, D., J. Stein, A. Otis, and A. Purdy, "Development of an Object-Oriented DBMS," *OOPSLA '86 Conference Proceedings*, Portland, OR (September 1986).

[Meag88] Meagher, R.B., McLean, M.D., and Arnold, J. (1988). "Recombination within a Subclass of Restriction Fragment Length Polymorphisms May Help Link Classical and Molecular Genetics," *Genetics*, 120: 809-818.

[McAr85] McArthur, D., P. Klahr, and S. Narain, *The ROSS Language Manual*, The RAND Corporation, N-1854-1-AF (September 1985).

[Mich87] Michiels, F., Craig, A.G., Zehetner, G., Smith, G.P., and Lehrach, H. (1987) "Molecular Approaches to Genome Analysis: A strategy for the Construction of Ordered Overlapping Clone Libraries," *CABIOS*, 3, 203-210.

[Mill91a] Miller, J.A., and N.D. Griffeth, "Performance Modeling of Database and Simulation Protocols: Design Choices for Query Driven Simulation," *Proceedings of the 24th Annual Simulation Symposium*, New Orleans, LA (April 1991).

[Mill91b] Miller, J.A., K.J. Kochut, W.D. Potter, E. Ucar, and A.A. Keskin, "Query Driven Simulation Using Active KDL: A Functional Object-Oriented Database System," *International Journal in Computer Simulation*, Vol. 1, No. 1 (1991).

[Mill91c] Miller, J.A., O.R. Weyrich, Jr., W.D. Potter, and V.C. Kessler, "The SIMODULA/OBJECTR Query Driven Simulation Support Environment," *Progress In Simulation*, Vol. 3, Leonard-Zobrist Editors (1991). (to appear)

[Mill91d] Miller, J.A., W.D. Potter, K.J. Kochut, A.A. Keskin and E. Ucar, "The Active KDL Object-Oriented Database System and Its Application to Simulation Support," *Journal of Object-Oriented Programming*, Special Issue on Databases, Vol. 4, No. 4 (July-August 1991).

[Mill90] Miller, J.A., W.D. Potter, K.J. Kochut, and O.R. Weyrich, Jr., "Model Instantiation for Query Driven Simulation in Active KDL," *Proceedings of the 23rd Annual Simulation Symposium*, Nashville, TN (April 1990).

[Mill89] Miller, J.A., and Orville R. Weyrich, Jr., "Query Driven Simulation Using SIMODULA," *Proceedings of the 22nd Annual Simulation Symposium*, Tampa, FL (March 1989).

[O'Bri90] O'brien, S.J. (1990) *Genetic Maps: Loci of complex genomes*, fifth edition, Cold Spring Harbor Press, Plainview, NY

[Olso86] Olson, M.V., Dutchik, J.E., Graham, M.Y., Brodeur, G.M., Helms, C., MacCollin, M., Scheinman, R., and Frank, M. (1986) "Random-Clone Strategy for Genomic Restriction Mapping in Yeast," *Proceedings of the National Academy of Sciences*, USA, 83, 7826-7830.

[Pear88] Pearson, W.R., and Lipman, D.J. (1988) "Improved Tools for Biological Sequence Analysis," *Proceedings of the National Academy of Sciences*, USA, 85, 2444-2448.

[Pott91] Potter, W.D., K.J. Kochut, J.A. Miller, V.P. Gandham, and R.V. Polamraju, "The Evolution of the Knowledge/Data Model," *Advances in Databases and Artificial Intelligence*, Petry-Delcambre Editors (1991). (to appear)

[Pott90] Potter, W.D., J.A. Miller, K.J. Kochut, and S.W. Wood, "Supporting an Intelligent Simulation/Modeling Environment Using the Active KDL Object-Oriented Database Programming Language," *21st Annual Pittsburgh Conference on Simulation and Modeling*, Pittsburgh, PA (May 1990).

[Pott89] Potter, W.D., R.P. Trueblood, and C.M. Eastman, "Hyper-Semantic Data Modeling," *Data & Knowledge Engineering*, Vol. 4, No. 1 (July 1989).

[Pott88] Potter, W.D., and R.P. Trueblood, "Traditional, Semantic and Hyper-Semantic Approaches to Data Modeling," *IEEE Computer*, Vol. 21 , No. 6 (June 1988).

[Pott87] Potter, W.D., R.P. Trueblood, C.M. Eastman, and M.M. Mathews, "KDL: A Hyper-Semantic Data Model Specification Language," *Proceedings of the 2nd International Symposium on Methodologies for Intelligent Systems, Colloquia Program*, Charlotte, NC (October 1987).

[Pott86] Potter, W.D., and L. Kerschberg, "A Unified Approach to Modeling Knowledge and Data," *Proceedings of the IFIP TC2 Conference on Knowledge and Data (DS-2)*, Algarve, Portugal (November 1986). (Published by North-Holland as *Data and Knowledge DS-2* (1988).)

[Prit86] Pritsker, A., *Introduction to SLAM II, Third Edition*, John Wiley & Sons, N.Y. (1986).

[Redd86] Reddy, Y.V., M.S. Fox, N. Husain, and M. McRoberts, "The Knowledge-Based Simulation System," *IEEE Software* (March 1986).

[Roth89] Rothenberg, J., "The Nature of Modeling," in *Artificial Intelligence, Simulation and Modeling*, L.E. Widman, K.A. Loparo, and N.R. Nielsen (Eds.), Wiley Interscience, N.Y. (1989).

[Rowl91] Rowley S., and Rockland, C. (1991) "The Design of Simulation Languages for Systems with Multiple Modularities," *SIMULATION*, 56:3, 153-163.

[Rudd91] Rudd, K.E., Miller, W., Werner, C., Ostell, J., Tolstoshev, C., and Satterfield S.G. (1991) "Mapping Sequenced *E. coli* Genes by Computer: Software, Strategies and Examples," *Nucleic Acids Research*, 19:3, 637-647.

[Rudd90] Rudd, K.E., Miller, W., Ostell, J., Benson, D.A. (1990) "Alignment of *Escherichia coli* K12 DNA Sequences to a Genomic Restriction Map," *Nucleic Acids Research*, 18:2, 313-321.

[Stan87] Standridge, C., and A. Pritsker, *TESS: The Extended Simulation Support System*, Halstead Press, N.Y. (1987).

[Ston86] Stonebraker, M., and L. Rowe, "The Design of POSTGRES," *Proceedings of the SIGMOD International Conference on Management of Data*, Washington, DC (June 1986).

[Suls88] Sulston, J., Mallet, F., Staden, R. Durbin, R., Hornsnell, T., and Coulson A. (1988) "Software for Genome Mapping by Fingerprinting Techniques," *CABIOS*, 4:1, 125-132.

[Zeigl90] Zeigler, B.P., *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, Inc., Boston, MA (1990).

[Zeig87] Zeigler, B.P., "Hierarchical, Modular Discrete Event Modelling in an Object Oriented Environment," *Simulation Journal* 49, 5 (November 1987).

[Zhan89] Zhang, G., and B.P. Zeigler, "The System Entity Structure: Knowledge Representation for Simulation Modeling and Design," in *Artificial Intelligence, Simulation and Modeling*, L.E. Widman, K.A. Loparo, and N.R. Nielsen (Eds.), Wiley Interscience, N.Y. (1989).