1 Introduction

This application addresses the continuation of research on the applications of artificial intelligence (AI) (1) to experimental molecular genetics. It is an extension of a longstanding effort to cultivate attention to ongoing laboratory research as a domain of explorations in artificial intelligence. Our major effort in this field had been in the DENDRAL project, with analytical organic chemistry as the object discipline. The present effort, "MOLGEN", focuses on a new object domain, namely molecular genetics. However, for reasons that will be elaborated, the focus is on programs to suggest experiment-planning sequences needed to solve a given structure, rather than on hypotheses about the structures themselves, which characterizes DENDRAL.

Our primary motivation is to deepen our knowledge of the art and science of creating programs that reason with symbolic knowledge to aid human problem solvers. The task domain--molecular genetics--serves as a rich intellectual and scientific environment in which to develop and test our ideas.

The major computer science issues we are addressing are:

Present Grant period (June 1976 - June 1978)

(1) Creation of a knowledge representation system with a knowledge acquisition package. The system, known as the Units Package, may be used to build a knowledge base in any suitable domain. It provides an object-centered approach for storage of both declarative and procedural information concerning all entities in the domain. Section 1.2.1

(2) Structured representation of process information. Procedures which simulate the action of the various processes in the domain form an integral part of the knowledge base. Moreover, the representation framework allows for inspection and acquisition of those procedures. Section 1.3.3.1.

(3) Creation of program schemata and instances for general problem solving steps. Domain-independent knowledge about general problem solving methods also fits into the knowledge representation structure we have devised. Section 1.3.3.3.

---

(1) For definitions of many of the genetic and computer science terms used throughout this proposal, see Appendix I.

(4) Domain Specific Critics. Mechanisms for the activation of various domain specific strategies when certain predefined situations occur during the course of experiment design. Section 1.3.3.4.

(5) Development of a specific planning strategy designed to provide high-performance for the class of genetic experiments known as discrimination experiments. The idea is based on indexing abstracted experimental designs to the types of structural features for which they have proven useful. Section 1.3.4.

Grant Renewal Period (June 1978 - June 1980)

(6) Creating a more comprehensive genetics knowledge base. Expanding the knowledge base within the area of DNA structural manipulation problems. This work will be done mainly by the Stanford Genetics Department. Section 2.1.

(7) Abstracting and Saving Plans. Recognizing when newly-created experiment designs are worth saving and then generalizing those plans so they are useful for more than the specific problem environment which caused their generation. This work will be done in the Computer Science Departments of both Stanford and UNM. Section 2.2.

(8) Making use of the process of hypothesis formation to help debug MOLGEN-produced experiment designs. This process is especially important in a domain like molecular genetics where incomplete knowledge about objects and processes is the rule rather than the exception. This work will be done in the Stanford Computer Science Department. Section 2.3.

(9) Experiment planning by analogy. MOLGEN provides an excellent environment for exploring various types of analogical reasoning. We integrate problem-solving by analogy into the experiment design system as one of the possible tools for solving subproblems. This work will be done in the UNM Computer Science Department. Section 2.4

(10) Performance evaluation as an integral part of the knowledge representation and acquisition system. We view the process of evaluating a system's performance and suggesting improvements as an AI problem solving task. The strategies for this evaluation will be

stored within the same framework as all other MOLGEN planning strategies. This work will done in both the Stanford and UNM Computer Science Departments. Section 2.5.

We have designed our effort to facilitate generalization to other domains beyond genetics in future research and applications.

A second motivation behind the proposed research is to develop tools that can benefit molecular geneticists. We believe there is substantial benefit to be derived from programs that act as "intelligent assistants" to scientists. First of all, the sheer amount of detailed knowledge a scientist is expected to know makes it likely that good experiments are being missed. Second, we believe that an intelligent planning assistant can offer some help in reasoning about the consequences of combining experimental facts in many possible ways.

A third motivation for applying artificial intelligence techniques to an experimental science like molecular genetics is to help us better understand the scientific method. The rigorous detail required for creating computer programs that assist in the performance of scientific tasks forces us to explicate concepts and procedures much more carefully than practicing scientists usually do.

During the current grant period, the basic MOLGEN system will be completed. (See Section 1.2 for details). This system will have a modest genetics and problem solving knowledge base, an ability to design a subclass of genetics experiments and a representation and acquisition system capable of handling all of the knowledge used by an experiment design system. During the renewal period we will expand the system's expertise and use the system to explore new problem solving capabilities. (Readers can skip to Section 2 for details of work to be completed during the renewal period). The basic MOLGEN system will provide an excellent framework for exploring more sophisticated experiment design activities and problem solving techniques. We have found in our analysis of the activity of geneticists that designing experiments is intimately coupled with forming hypotheses about the failure of the experiment design in the laboratory, then testing the hypotheses and, as a result of the tests, gaining new knowledge wtich effects the design. The new hypothesis formation component of MOLGEN will extend problem solving capabilities in this direction. Another important element in the repertoire of an expert human problem solver is the ability to form analogies between problems and to use the analogy to solve new problems. Our exploration of analogical reasoning within the MOLGEN system will lead to a new problem solving component integrated with the rest of the design system. New expertise will come in a direct manner from the expansion of the knowledge base by geneticists. At the same time we will increase our problem solving knowledge base by adding a component which monitors the problem solving

experience of the system and abstracts new experiment designs for the design process. The generalization process developed for this monitor will be used by the analogical reasoning component. Our final extension to the MOLGEN system is the ability to evaluate its own performance. This evaluation will aid the development of all of the other extensions.

1.1 Motivations

The successful initiation of and perseverance with interdisciplinary research of this kind can be hindered or aided by many intangible factors in the local context, as well as by the personal idiosyncrasies of the participants. We need only point to the history of the DENDRAL project as evidence of our proven capability, for whatever reasons, to engage in such research in the Stanford University environment. It is inevitably true that no one person can embrace a professional level of critical insight over all the fields represented, but this has been surmounted by dint of hard work on mutual communication from all sides. More problematical is the problem of communicating our findings to a disparate group of disciplinary experts outside, e.g. to computer scientists and to molecular geneticists who customarily have had little need of perusing each other's disciplines. The best solution is to get these parties engaged not just in reading proposals or publications, but in the actual manipulation of the working programs -- particularly with the help of computer networks that link dispersed sites. Even so, an amount of technical detail that could well try the patience and skill of the most tolerant reviewer is unavoidable if we are to offer a concrete, fair picture of the status of our efforts, especially in the formative stages. We trust that, as is often true, the maturation of these efforts can be accompanied by a simplification in presentation, and meanwhile we must appeal to the patience and good will of constructive critics. The supporting documents attached to this proposal offer a sample of our working papers and publications. Further documentation on any aspect of the present system is, of course, available.

1.2 Progress To Date

The following sections describe our progress towards developing an experiment planning program. The first part discusses a knowledge management system. Most of our design and programming effort has been in this area. A package of programs for the representation and acquisition of knowledge is discussed and our progress in using it in creating a genetics knowledge base is also described.

The next section focuses on our examination of genetics knowledge. It discusses various studies we have made of genetics experiments and what we have learned from them. A short section describes some work we did in representing the action of the ligase enzyme and how this was of benefit to work in Prof. Lederberg's laboratory.

1.2.1 Knowledge Base Research

The success of MOLGEN as an experiment planner will depend on the quality of its knowledge base. Therefore, much of the research effort to date has been in the design and implementation of a knowledge representation and acquisition system. All of the information relevant to the planning process will be an explicit part of the knowledge base. The motivation for this aspect of the design is the necessity to expand the program capabilities in a modular fashion and to explain the rationale behind the program's planning behavior. We need to represent concepts (e.g. enzyme), instances (e.g. EcoRI), relationships among concepts, and relationships among instances. In addition, we need to represent processes. Two recent papers [42][3] have delineated problems which arise in interpreting a knowledge base when the semantics of the representation are not clearly specified. We have purposely limited the expressive power of our representations to enable us to clearly define their semantics.

The result of this work is the Unit Package. Although this package has been designed in the context of our genetics application, the package does not contain any genetics knowledge. At this time one other group within the Stanford Heuristic Programming Project, the AGE group, is using the package.

Our knowledge base design has been reported in the literature [24]. Other documents relating to its use are included in the appendices. Also included is a sample session illustrating the acquisition of a DNA structure and a current summary of part of the knowledge base dealing with enzymes. Section 1.2.1.1, Section 1.2.1.2, and Section 1.2.1.3 describe the implemented portions of the representation package. Section 1.3.1 discusses some of the further developments for the knowledge base that are planned for the rest of the current grant period.

1.2.1.1 The Unit Package: Background and Relation to Other Research

Design of the representation package (termed the Unit Package) was started in November 1976 although most of the programming has been done since March 1977. The design of the Unit Package profited from

the experience of other efforts e.g. KRL-0 [1], SMALLTALK [20], and the work at the MIT AI Lab on frames [27][21]. One important difference in perspective between the Unit Package research and many other efforts is that it is at once a knowledge acquisition system and a knowledge representation system. The knowledge acquisition system is in the same spirit as Davis' work on TEIRESIAS [7] where schemata were used to guide the acquisition of knowledge. Another distinguishing feature of the Unit Package is that it is also used to represent and acquire process information, that is, action and strategy knowledge (see Section 1.3.3.1). The package performs many other knowledge management tasks for the system.

The basic element of our representation is an entity called a unit. Units are either "prototypes" or "instances". Prototypes are used to represent Woods' intensions [41], Brachman's concepts [3]. That is, a prototype defines a class, that is, the knowledge expected for a particular concept in terms of slots with the appropriate fields (e.g. role, value/specification, default). (2) Individuals are defined as instances of prototypes. Subclasses can be formed with a special link called generalization/specialization. Concepts can only have one generalization and instances only one prototype. Other relationships among units can be expressed by the value/specification of a slot or by creating an explicit relation unit.

The knowledge acquisition system is used to acquire both prototypes and instances. This system is being used by system builders to create the bootstrap network described in the next section and by Jerry Feitelson, our genetics research assistant, for defining and instantiating knowledge about several families of enzymes. (See Appendix II and the supporting documentation for examples of knowledge acquisition.)

Other knowledge base management chores include automatic bookkeeping and knowledge integration. The bookkeeping chores simply keep a record of all pertinent information about the creation or modification of a unit.

1.2.1.2 The Bootstrap Knowledge Base and Object-centered Perspective

One important aspect of the design of the system is that the knowledge base contains knowledge about its own data representations. We have provided what we term a "bootstrap knowledge base." It contains domain independent knowledge about commonly used data types. When

---

(2) The slots in a prototype are similar to Brachman's DATTR links. A limited version of Brachman's MODALITY link is implemented. MODALITY specifies the necessity of the specific DATTR in instances.

using our knowledge base in a new domain, an artificial intelligence researcher would probably start with the bootstrap knowledge base and then proceed to create units for the specific knowledge of his task area. Both the AGE and genetics knowledge bases have been started in this manner.

The bootstrap knowledge base serves to illustrate our approach to extensibility. Most of the bootstrap knowledge base is made up of the procedures which capture the knowledge in the system about primitive datatypes. To add a new datatype to our system, one needs to provide the knowledge base with procedures for some basic operations -- such as editing and printing. Actually, the same approach is used in the unit package for defining a new datatype as is used for defining a new enzyme. The process of defining new datatypes requires, however, an understanding of Interlisp because the primitive processes in the system are grounded in that language. New datatypes must be defined together with their basic operations and entered into the knowledge base.

The approach used for attaching the basic operations to the primitive data types is very similar to SMALLTALK [20] and KRL-0 [1]. As far as the knowledge base is concerned, in order to edit a DNA structure, one sends the DNA unit an edit message. Another way of saying this is that the procedure for editing a DNA structure is attached to the DNA unit. Thus, indexing of the "edit procedure" is via the "DNA object" so that this is an application of the object-centered viewpoint introduced in the languages mentioned above. This organization makes it easy to add new datatypes to the knowledge base.

An object-centered viewpoint is used in our system for other applications as well. An example of this is the "inspector" idea discussed in Section 1.3.3.4 which illustrates the significance of object-centered indexing during planning. In that example, particular knowledge about related planning situations is associated with the genetic object -- "pH". Generally only operations basic to primitive data types are represented in LISP code. Other process information is expressed as units. Instances of process units can also be associated with objects.

1.2.1.3 Considerations of Human Engineering

Because we expect our knowledge base to be maintained by geneticists, it has been important to carefully consider the interaction expected between a geneticist user and our Unit Package. This effort has resulted in the crafting of a package which is particularly easy to use. This ease is verified by the acceptance of our system by both geneticists and users on the AGE project.

One part of human engineering has already been mentioned above. The system automatically records who has entered or modified any of the information in the knowledge base. It provides summaries of what is in the knowledge base and can indicate which areas of the knowledge base have been changed recently. (See the attached supporting documentation for a summary of a recent version of the genetics knowledge base.)

Another example of this human engineering is the "User Profile" maintained in the unit editor. A user's profile contains such information as whether he is a programmer, whether he prefers verbose printouts, whether he likes to be informed when the system does things automatically for him (i.e. that he has not requested). This information helps the system to tailor its interaction with a user. In addition, terminal communication is handled via very general routines which check for type ahead and allow help information to be provided to the user at any point in the dialog. The result of this is that an experienced user can have very brief interactions with very little prompting from the system. An inexperienced user will be prompted at each step of the dialog and may type "?" at any time to get an explanation or example of what the program expects. (See the User's Guide to Units in the attached supporting documentation.)

Many of the human engineering parts of the system take advantage of the facilities of InterLisp. For example, when a programmer is changing some procedures in the knowledge base, he need not explicitly request that the changed functions be recompiled. This part of the updating is performed automatically at the end of a session.

Our final example of human engineering illustrates the fuzzy borderline between the technical problems of making the system easy to use and some more complicated issues of knowledge base management. Recognizing the importance of experimentation in representing knowledge, one of the facilities of the unit system is to merge knowledge bases. This makes it easy to update various knowledge bases when changes are made -- for example when the bootstrap knowledge base is updated. Part of the transfer protocol includes the facility for one unit to imply a requirement for another. For example, suppose that the bootstrap knowledge base has just been updated so that it has knowledge about the representation of an important data structure - say "trees". If this is useful in representing knowledge in the AGE system, they need only request to transfer the "tree unit" from the bootstrap knowledge base. Any other units essential to complete the knowledge about "tree representation or tree operations" will be transferred automatically at the same time.

The more difficult aspect of merging knowledge bases concerns how the system can determine which units should be transferred when a particular unit is transferred. Restated -- how can the system find

all of the relevant knowledge about a unit? Our approach to this involves a method of inserting knowledge about relevance into the knowledge base itself.

Considerations like those in the previous paragraph are simple enough from the implementation point of view and are described in the appendices. They are representative of the effort we have taken to make the units package easy for a geneticist (or a programmer) to use.

1.2.2 Studying the Process of Experiment Design in Molecular Genetics

Considerable effort was given during the first fifteen months of the MOLGEN grant to the study of the process of designing experiments. Attempts were made to determine subsets of the domain which would provide realistic target areas for automated experiment design and to collect the strategies and domain knowledge used by experts for those subsets. Detailed descriptions of the results of this investigation are given in the appendices.

The work included four major efforts:

1. Designing as many rational experimental plans as possible for an experimental problem from Prof. Lederberg's laboratory, the determination of the presence of poly-A sequences in DNA. (See attached supporting documents: "The Embedded Sequence Problem")

2. Collecting "skeletal" or abstracted plans for a wide variety of structural elucidation problems. These plans are meant to capture the re-usable aspects of an experimental plan for use in planning other experiments. They are an essential component of an approach to planning described in Section 1.3.

3. Analyzing in detail the logic and development of a single experiment performed in Prof. Lederberg's laboratory. This study considers both the knowledge which led to the successful experimental design, and also the process of hypothesis formation to account for experimental failures. (See attached supporting documents for a copy of [14])

4. Analyzing a collection of experiments suggested by several geneticists as good examples of logically produced experimental designs in order to extract the knowledge needed for MOLGEN to design the experiments.

Our design of the system has been strongly influenced by these studies. Rather than give detailed results in the body of this proposal, we present a summary of these influences.

1. We will concentrate our initial efforts on experiment planning to the subfield of restriction enzyme experiments and then expand to applications to plasmid and sequencing experiments.

2. We will represent genetic objects so that they may be viewed from a variety of different perspectives. Our studies showed that different perspectives led to different experimental algorithms. (See [40] for a similar finding in program synthesis.)

3. While we are restricting the range of objects and processes as indicated in (1), we must represent many different types of knowledge about each object. Even in a single experiment, a great diversity of knowledge was needed to design the experiment [14].

4. The planning system will be one component of an eventual genetics expert. The experimental design process is much more event-driven than anticipated. The eventual expert system will need a hypothesis formation component and techniques to exploit serendipity. For example, results of a particular transformation may indicate interesting sidelights which could be investigated in the context of the current experiment.

1.2.2.1 Simulating the Process of Enzymatic Ligation

An attempt was made early in the first year of the current grant to simulate the action of enzymatic ligation on DNA structures. There were several motivations behind this effort. We wished to test our newly developed DNA structural representation and to learn something about representing the processes that modify those structures. Also, we were presented with an interesting problem where we could be of actual use to a laboratory geneticist (Dr. S. D. Ehrlich).

The experiment in question was one in which sticky ended DNA of uniform length was reacted with ligase to join together ends. Each ligation could either join two different DNA structures or could circularize a single structure. In the former case the new structure would still have two sticky ends and the total number of structures in the sample (concentration) would be reduced by one. In the latter case

the number of structures would remain the same, but one structure would be "inactivated" for further ligation. The problem was to predict what the total sample population would look like after some portion of the original monomeric structures had disappeared. That is, what percent of the total structures would be monomeric circles, dimers, dimeric circles, and so on. These percentages would be compared with laboratory results in order to determine how many different ends types were present in the sample. (The more different end types, the greater the ratio of circles to linear and shorter structures to longer structures).

The simulation we developed was based on a kinetic theory of ligation [11]. The simulation results correlated remarkably well with the actual laboratory experiment when a model with two different end types was chosen. This gave additional support to Dr. Ehrlich's conclusion that indeed two different structure types had been present. The program was then used for the related experiment of maximizing inserts of a small gene into a plasmid. The idea was to cut the plasmid with the same restriction enzyme that had produced the sticky-ended gene, join the two, and then recircularize the structure to produce a recombinant plasmid. The problem is that many other structures, monomeric circles, long linear concatemers, etc., can occur. The problem was to determine optimum starting concentrations of the two structures and the length of time to carry out the ligating process to maximize the desired product. The suggested values given by the simulation program were of considerable assistance in carrying out the laboratory operation.

The major significance of this work for future MOLGEN development (besides giving a working geneticist assistance and encouraging his cooperation in communicating problem-solving knowledge to us) was that this was our first attempt to capture the knowledge needed to simulate a genetic process. We made no effort to consider environmental side effects like temperature and pH, but the results were still quite encouraging. The major problems we discovered were twofold and related. First, the simulation program was hard to expand to other problems. Adding other relevant information, for example stereochemical effects, or about new processes such as cutting structures, would involve considerable additional programming. Second, all of the knowledge was hidden in the simulation program (written in SAIL). The program could offer no explanation of its predicted results, nor could the user make interactive changes to its guiding theory. These results led to our decision to make all knowledge, not just knowledge about static genetic objects, explicit in our knowledge base. A discussion of how this is being accomplished is given in Section 1.2.1 and Section 1.3.3.

1.3 Plans for the Remainder of this Grant Period

1.3.1 Continuing Knowledge Base Management Research

Our progress in developing a knowledge representation system for MOLGEN has been discussed above. Our continuing effort during the grant period is divided into (1) filling in a knowledge base with entries specific to a specialized subset of molecular genetics, and (2) expanding our representational capabilities to include knowledge about the process of experiment design.

1.3.2 A Genetics Knowledge Base for Restriction Enzyme Experiments

The Units package described above will be used to acquire and store the knowledge needed to plan experiments. The initial knowledge base will be limited to the facts and heuristics necessary to reason about restriction enzyme experiments. This will include:

1. Units for basic concepts like DNA structure, enzymes, and samples.

2. Units for each individual restriction enzyme.

3. Units for laboratory techniques which make use of restriction enzymes.

4. Procedures which describe the action of restriction enzymes on DNA.

After the restriction enzyme knowledge has been debugged, knowledge about the fields of plasmid technology and sequencing DNA will be added.

1.3.3 Knowledge About Processes and Planning

The basic planning work in MOLGEN will be divided into two major efforts. General domain-independent planning strategies will be developed within the units knowledge representation system as part of the process of making an "AI toolbox." At the same time, work will proceed on a specific strategy tailored to provide high-performance for discrimination experiments in molecular genetics. The specific approach for discrimination experiments is discussed below after the discussion of our general approach to process representation and planning.

1.3.3.1 Representation of Processes

Extending the Unit Package to accommodate the representation of processes is the important next step in the development of the knowledge base system. We view both simulation (of a laboratory technique) and planning operations (e.g. selection of a subgoal) as processes. Our approach to the representation of processes has previously been discussed [24][36]; the main ideas are reviewed below.

We want to provide a facility that makes process description and debugging as easy as possible. For example, the system will provide an appropriate set of primitive operations for the operations on DNA models. The geneticists and the computer scientists will create prototypes for laboratory steps. Geneticists will then use these prototypes as guides during the acquisition of knowledge about particular laboratory tools. The same prototype/instance system will be used to represent planning operations.

Many of the processes we want to represent can be grouped into classes such that there is a prototypical action for each class. That is, the individual processes (e.g. action of EcoRI) can be viewed as instantiations of a process concept (e.g. action of restriction enzymes) just as individual objects are the extensions of object concepts. Many researchers in program synthesis (such as [9]) have used a "program schema" to represent a prototypical program. The schema is then instantiated by the synthesis system to produce a concrete program. A program schema consists of a generalized program with abstract predicate, function and constant symbols, input/output specifications, and restrictions on possible instantiations of the abstract symbols.

This notion of a program schema (we will use the expression "process schema" synonymously) can be implemented within the current unit representation. The abstract predicate , function, and constant symbols are slot names. The restrictions on possible instantiations are specified in the value/ specification field. The abstract function symbols may be restricted to be either system primitive functions or instantiations of other program schemata. I/O variables and relationships among them can also be specified via the slot mechanism and the "relation" units. Within a rule unit there is a special slot which contains the generalized procedure. Thus, a program schema can in fact be developed in a modular fashion: first create the simpler, or primitive schemata; then decide on the constant, predicate, and function symbols needed (i.e. the named component parts in a process description); then create the generalized procedure using these symbols as the named components.

Implementation details of rule units are currently being examined. While in synthesis programs, the system creates the

instantiation from program segments, in the current design of MOLGEN, the user creates the instantiation. The acquisition system must be able to check instantiations to make certain that the restrictions and I/O specifications have been met.

The use of process schemata and their representation as units has implications for planning also. The visibility of process components in slots is important so that other processes can examine and select them according to the values in the slots. General problems of such indexing methods and comparisons of systems which use these methods can be found in the literature [8][36].

In many cases it is appropriate to associate the process elements of the system with definitions of the objects. This "object-centered" viewpoint has been expounded [1] and implemented as "attached procedures". Several different applications for attached procedures in MOLGEN are given [24]. These attached procedures can be system primitives or instances of process prototypes.

1.3.3.2 Representation of States and the Planning Network

Any system for doing problem solving needs to be able to work with representations of domain worlds (often termed "world states") and representations of problem solving states. An example of a world state in MOLGEN is a sample which might contain a variety of hypothesized DNA structures, enzymes, and other reagents. An example of a problem solving state is a sequence of laboratory steps and world states which represent several steps of an experiment. We find it convenient to consider a problem solving state as having levels of detail, with different kinds of problem solving information on the different levels. A description of our approach using such levels is given in [36], section V.3.

Two main ideas have influenced our design. First, the world states and planning states are fairly complicated structures. It is necessary for a planning program to communicate with its users about its current planning state. In particular, it must be able to display its progress in an easily understandable form and it must be able to integrate suggestions from the user to alter its course. Rather than create separate programs for doing this, we have decided to represent the planning and world state knowledge using units in the knowledge base.

The second idea is that the process of problem solving can be expressed adequately by a small number of basic operations. These operations are used repeatedly in the course of solving a complicated problem. This is discussed in the next section.

1.3.3.3 An Eclectic Perspective on Problem Solving -- The AI Toolbox

When we study the solution to a problem in an unfamiliar domain, the first reaction is to be overwhelmed by the new detail and terminology. When one has mastered the terminology, the problem solving process can be viewed in better perspective. To be sure, many solutions remain brilliant and surprising. The majority of the solutions are easy to follow and we may recognize the solution process in the mind of the problem solver. Here he is sketching out some goals; now he is selecting an operator; now he is refining a step; now he has factored out a subproblem.

We believe that the process of problem solving can be modelled by a small number of standard operations. These operations include such things as (1) Sketching out planning islands; (2) Proposing subproblems; (3) Testing for mismatch of goal states; (4) Focusing attention on part of the problem; (5) Assigning time sequence to steps; (6) Selecting among competing choices for a given step; and (7) Splitting a problem into cases. Although these have been recognized as basic components of strategy, no existing system has integrated this breadth of strategy knowledge into a knowledge base. Dershowitz and Manna [9] have suggested the use of program schemata to represent general problem solving techniques such as Divide and Conquer. These program schemata are instantiated by the program synthesis system to obtain concrete programs. In our system the geneticist and/or programmer will create the instantiation. A major component of this research will be in the construction of such program schemata and instantiations. The package created will be termed the "AI toolbox". The creation of a library of schemata has been suggested by Gerhart [19]. The package will be built and tested in the context of a small number of genetic experiments. Computer science experiments with the package will include a measure of its performance and utility for expressing appropriate strategy for experiments.

Of course, having a number of basic tools does not guarantee that a program will use them correctly. Competence in problem solving in the domain requires that the expert system know where and when to apply the standard methods. Two factors are of importance here. (1) Some tools govern the use of other tools. For example, "Focus processes" will determine where to concentrate effort in a problem and indirectly control the selection of strategies. (2) The specific knowledge for using a tool is precisely the type of information which is left unspecified in the prototype and which must be supplied when concrete instances are acquired for the knowledge base.

The AI toolbox is discussed further in [36] in section V.4.

1.3.3.4 Simplifying Process Specifications by Removing Exceptions

Planning can be plagued by exceptional cases. If high level planning processes are burdened with the detail of the special cases, they can become cumbersome to update and debug. We are developing an object-centered approach, termed inspectors, for distributing the information about special cases throughout the knowledge base. This should help keep the planning processes clear and concise and also provide localized packets of information about the exceptions.

The basic ideas of this process of removing exceptions can be illustrated by an example of a "selection process" in designing an experiment. The operation of selecting among available laboratory steps is a recurring operation during experiment design. The following bear on this process:

1. The experimental goals -- e.g. to extract a section of a molecule.

2. World State specification -- a description of the laboratory sample, e.g. DNA structures.

3. The selection criteria -- e.g. availability, sensitivity, or functionality of the laboratory technique.

4. Verification criteria -- test for deciding after a simulation of the selected laboratory step whether the essential goals have been satisfied.

5. Failure instructions -- what to do if the chosen laboratory technique does not satisfy the goals.

6. Laboratory step specifications -- a list of the potentially applicable laboratory tools and their descriptions.

Our approach to managing the information about a selection process is first to identify the general information and then to consider the alternatives for placing information about the special cases. For example, section V.4.4 of [36] considers the selection of an enzyme to make cuts around a region in a DNA molecule so that it may be extracted. The general information in this case is that the main determinants of selection are enzyme availability and information about where it will make cuts. Special case information includes modifications on this basic idea according to unusual variations in the structure of the molecule -- e.g. "AT-rich regions", "hairpin loops", etc. It also includes information specific to the laboratory steps being selected -- e.g. nuclease contamination in an enzyme.

Interactions between goals often arise in the special cases (3). When action was taken to satisfy the preconditions of the enzyme (in this case, pH was changed), a side effect resulting from interaction with an unusual molecular feature interfered with the action of the enzyme. Specific advice about failures of this kind can be associated with the "pH inspector" -- To avoid conflicts of this kind an enzyme should be used at a suboptimal pH. Inspectors may be viewed as domain specific versions of planning critics as developed by Sussman [37] and Sacerdoti [32].

Generally there are many tradeoffs involved in deciding where to locate the information about special cases. The example above and the tradeoffs are described in detail in section V.4.4 of [36].

1.3.3.5 Further Aspects of Knowledge Base Management

One further area of work in knowledge base management that we will be pursuing is in developing a number of modest aids for a user for keeping track of a growing and evolving knowledge base. As discussed previously, we have already developed some facilities for automatic documentation of the knowledge base. We will be designing aids for a user in tracking down unexpected conflicts. For example, if two geneticists sharing a knowledge base have a somewhat different view of some aspect of the domain, one may make changes effecting the other's area. In the event of a failure, it would be useful in many occasions to get a summary of recent changes to the knowledge base. When a change is contemplated to the definition of some class of enzymes, it would be a simple matter to locate all of the rules which mention those enzymes. These rather simple aids are expected to be useful when fairly extensive changes to the knowledge base are contemplated because they will assist the user in being thorough about making his changes.

1.3.4 A Method for Designing Discrimination Experiments

One of the applications of the genetics knowledge base will be the building of a high performance system for designing a variety of discrimination/analysis type experiments. The goal in these experiments is to learn something about a given sample of DNA structures. For example, are any poly A regions present, or do the structures carry tetracycline resistance? The basic method used for the experiment design system will be means-ends analysis combined with hierarchical planning as follows:

---

(3) See Section III.2.5 of [36] for a discussion of recent techniques for handling interactions between goals.

1. A model structure containing the hypothesized feature(s) will be compared with a structure representing generalized DNA of the same type in order to ascertain exact differences.

2. These differences will be ordered and one selected as a basis for initial planning.

3. An experimental strategy, ranging from very specific (e.g. if a bubble is present then denature and use EM) to very general (e.g. label the feature and then look for the label) will be selected from a library of such "skeletal plans," using the difference selected.

4. The skeletal plan will be adapted to the specific problem environment, with hierarchical planning proceeding as deeply or shallowly as is desired by the system user.

5. The completed design will be tested in a forward direction for completeness and consistency by an evaluation system.

6. If a successful design cannot be found from the feature selected in (2), then either a new feature will be selected from the difference list, or a generalization selected from a tree of structural features will be used. For example, if the selected feature was the exact base sequence ATTGC, a generalization might be made to "known base sequence."

The following major components are needed for this experiment design method:

1. A problem analysis preprocessor which recognizes key features in the nucleic acid structures, nicks, gaps, hairpins and the like, and then compares these structural features to find major differences between candidates in a discrimination experiment. This program will organize features into a hierarchy of importance for experiment design using a rule-based system for analysis and classification.

2. A tree structure for ordering structural features and providing links between features and the classes to which they belong. The links will point upward as generality links to a more general class or downward as specificity links to a more specific one. The highest links will be to the complete class of

structural features, the lowest ones to individual specific features. Intermediate levels will consist of important subclasses, e.g. "poly base sequences" which would point downward to "poly A," "poly T," "poly G," and "poly C," and upward to "AT/GC ratio", which would itself point upward to known base sequences." This tree will provide entry into genetic strategies classified by the features to which they specifically pertain. Failure to find knowledge about what to do with a specific feature will cause the system to search for knowledge about the subclass of features immediately above the specific feature in the tree.

3. A cohesive system for hierarchical planning in the domain of molecular genetics--selecting strategies as described above and refining them downward to specific laboratory tools. This of course involves the usual need for error recovery and backtracking facilities.

4. A system for evaluating completed designs at any level of generality to be sure the plan as a whole "fits" togethers, i.e. a forward-working system for plan evaluation.

These components will be integrated into the basic MOLGEN representations framework of the Units system described above. The rules describing the problem analysis preprocessor will be individual units, as will the descriptions of structural features which combine to form the tree structure of the feature hierarchy. Individual dynamic planning states will be represented as units within the MOLGEN system.

After the experiment design system becomes operational for discrimination/analysis type experiments, we intend to adapt it to experiment-planning for synthesis experiments where the goal is to produce some desired DNA structure from available starting materials.

1.4 An Example of the Genetic Utility of Automated Experiment Design

A good way to illustrate the potential utility of computer assistance in experiment design is to show how some recently published work from another laboratory might have been represented in a MOLGEN formulation, albeit human intelligence was the instrument. The work in question achieved the cloning of the gene for rat insulin in a bacterial plasmid vector [38]. The major goal of the experiment was the transfer of a gene coding for insulin from the rat to the common

intestinal bacterium, E. coli. An important subtask of the experiment was: given samples of two different linear DNA structures with "sticky" ends, produce a circular structure containing one molecule of each. The difficulty of this problem lies in the number of competing processes. Both structures can self-circularize, and many different linear and circular monomers can be produced. See Appendix III for further details.

Previous attempts to cope with the problem were based the kinetic theory of ligation [11]. Using a model of the process based on concentration and molecular weight of the structures, one varied various experimental parameters to maximize the amount of the desired product. The new idea of the article was based on a different strategy--try to eliminate competing processes. This led to a method for modifying the sticky ends of the two structures so that they could no longer self-circularize. The particular method chosen was a simple biochemical step. The solution was retrospectively self-evident, but in fact, it was missed by many geneticists who had previously examined the problem (or related others). An intelligent experiment design system with the above mentioned heuristic probably would not have overlooked the solution.

2 Research Plans

The bulk of this application comprises details of research strategies for the first two years of the renewal period. It is of course more difficult to forecast over longer periods; indeed there is every likelihood that unforeseen difficulties and opportunities will intervene to offer changes of perspective. However, enough progress may have been made to enable us to move from the initial stepping stones, and our current plan for years 3 and 4 is as follows:

We will by then have invested substantial effort (mainly through the cooperation of investigators with direct support for molecular genetics research) in building and maintaining the knowledge base in the specified domain. We believe that this investment should be exploited before substantial further efforts are made to expand the domain -- e.g. in more biological aspects of genetics -- although a few easy opportunities will surely present themselves. Instead our emphasis will be on the evolution of MOLGEN to a hypothesis-oriented system (like DENDRAL). Typical questions at that level would be: from the data given, what are plausible hypotheses for the structure of a sample of DNA; as well as, what experimental steps should be pursued to verify the hypothesis. Intermediate steps have already been mentioned: the elaboration of hypotheses in the course of debugging, and the extension of MOLGEN from a sharp binary discrimination (between two

stated alternatives) to a corroborative mode (a given hypothesis versus all plausible alternatives).

This effort will require a good deal of work on the planning components and on rules of plausible induction, and relatively little on the knowledge base of molecular genetics per se. For that reason it should have the greater value for extensions to other domains.

2.1 Building and Maintaining the Genetics Knowledge Base

One important building block for MOLGEN's success is the creation and updating of a base of knowledge about genetics. This provides both a useful reference source for the user planning his own experiments, and a body of core data on which automated experiment planning strategies can be tested and evaluated.

The knowledge acquisition and representation ideas have been delineated in detail previously. This section describes advantages of the resources available here to support this work.

The strength of the Stanford community of biologists, biochemists, and geneticists offers a unique opportunity for collaboration in building a substantial body of knowledge about on-going research projects. Many of these projects may prove to be suitable sub-domains for initial development of the knowledge base, and may provide test domains for work on problems of knowledge acquisition and knowledge base management. They offer the added advantages of insuring that this work is solidly grounded in real-world experiments.

Updating and checking the knowledge base with respect to the information obtained from our collaborators will be the responsibility of the graduate student and post-doctoral fellow in genetics. On oppornunistic occasions they will also experimentally validate those experiment plans generated by the system which are relevant to current work in the lab. This has already been done in the case of ligation kinetics, where theoretical predictions of plasmid self-circularization were compared to electron-microscopic empirical tests (Section 1.2.2.1).

One useful side-effect of this process that we anticipate is that the process of formalizing knowledge about the domain may help to organize what is currently an informal body of knowledge, and in doing so may even uncover gaps in our current store of knowledge about the field.

2.2 Recognizing and Abstracting and Saving Successful Plans

As noted above, our efforts to create an experiment-design system center around the concept of a large knowledge base containing task-specific information. One approach to augmenting this knowledge base is via interactive knowledge acquisition as discussed in Section 1.2.1.3. A second form of knowledge-base improvement is based on giving the system the ability to save successful experiment designs it has generated. This involves two major functions: recognizing when a plan is worth saving, and abstracting a plan so that it is applicable to a wider range of problems than just the specific one which prompted its creation.

For example, consider the design MOLGEN might produce for the problem of ligating two genes. An initial attempt would be made to produce "sticky" ends by cutting both genes with a single restriction enzyme. Suppose, however, that no restriction enzyme which satisfied this criteria could be found. One possible solution would be to cut the DNA near one gene with restriction enzyme A, and near the other with a different restriction enzyme B, then join the two segments by means of a small piece of artificially created DNA which had been cut on one end by A and the other by B. This general idea -- the concept of a "molecular adapter" [35] -- is very useful for problems of this sort. While recognizing and abstracting the relevant ideas from the specific experiment design above is difficult, the ability to do this would be an important form of knowledge base augmentation.

Automating the function of recognizing when a plan should be saved will rely on several interrelated factors:

1. Was the plan a "good" one, i.e. does it solve a problem with reasonable efficiency, cost, safety, etc.? This measure of "goodness" will be difficult for the system to judge alone. The system will rank alternate plans according to its heuristics which involve these factors, but an absolute measure of plan quality will be a very difficult measure. Until the knowledge base includes adequate metrics for the "goodness" of plans in a variety of subfields, we imagine this judgment will be mostly up to the user.

2. If the plan solves a problem for which other plans already exist in the knowledge base, is the plan a significant improvement? Making this judgment involves use of the same "goodness" metrics discussed above as well as an analysis of why the new plan was not simply a copy of the old. If the only difference between the two plans is that the new one takes account of some detail of the environment for a specific problem, then it probably isn't worth saving. Again, we imagine this judgment will, at least initially, be mostly up to the user.

3. Is the cost of saving the plan less than the cost of regenerating it each time the same problem arises? Empirical criteria applicable to making the tradeoff decision might include CPU time spent in generating the plan, and, if the plan evolved directly from a previously saved one, the number of differences between the two plans. Another important measure is some judgment about how often the plan will be useful in the future. The more generally useful a plan, the more important it is to keep it around.

4. Can the plan be abstracted to more general purposes? In the example given above, this would involve recognizing that the concept of a "molecular adapter" would be useful any time the goal was to join two pieces of DNA which did not have convenient restriction sites in common.

The problem of abstracting a plan that has been selected for saving has been considered before, in the robot planning domain, in the work on MACROPS in the STRIPS system [15]. STRIPS generalized successful plans in the following manner. Plans had variables which were bound to objects in the robot world. The idea was to "unbind" these variables as much as possible, with certain constraints imposed by the nature of the particular plan. For example, if one step of a plan told the robot to go to a block and the next step said pick up a block, the block in these two steps must be the same one. Initial references to specific doors, blocks, rooms, etc. were generalized to "any door," "any block," "any room,", etc., but took into account constraints of the sort noted above.

We will begin by employing much the same generalization process. For the example plan given above, the specific restriction enzymes could be generalized to any two distinct restriction enzymes, and the two genes to any two DNA sequences lacking a common restriction site. This generalization process should be a natural consequence of the hierarchical nature of our knowledge base. The taxonomical classification of DNA structural features discussed above (Section 1.3.4) could be used to generalize plan utility--e.g. if a plan solves a problem for nicks, maybe it can be generalized to solving that problem for the immediately more general parent of nicks, DNA excisions. A plan which was useful for recognizing a specific base sequence might be generalized to one which was useful for all "known base sequences." The generalization of variables within a plan will be done by moving up generality links (see Section 1.2.1.1) in the knowledge base. A particular exonuclease could be parameterized to mean any exonuclease--the parent of all specific exonucleases in the knowledge base.

One difficulty lies in knowing how far it is possible to go up the generality links in the knowledge base before losing plan utility. A specific enzyme could be unbound to refer to "any enzyme," but that

would yield a plan which said "pick any enzyme", and would probably be far too general. The problem lies in detecting the important (i.e. important to this particular plan) features of each object used in the plan, and trying to retain those features while generalizing out all "irrelevant details." In the plan discussed above, the important feature of the chosen restriction enzymes is that they cut at specific sites, not that they were derived from some particular organism or that they operate optimally at a certain temperature. Note that such information (the degree of relevance of the features) will be available from the hierarchical planning phase, since it will have been used to choose among competing laboratory "tools". In this case, for example, restriction enzymes were chosen because they cut at specific sites. So, in unbinding the notion of a particular restriction enzyme, if the knowledge of the importance of specificity is maintained, progress up the generality links will end at "restriction enzyme."

2.2.1 A Casebook of Unsolved Problems

A related area of investigation -- recognizing novel combinations of laboratory techniques -- is based on the observation that new tools are continuously being developed. Sometimes a particularly useful combination of tools is available for quite some time before it is recognized as such, as in the use of alkaline phosphatase to inhibit self-ligation mentioned in Section 1.4.

One approach for doing this is to keep on file a casebook of important or unsolved genetic problems. Periodically as new laboratory techniques and planning strategies are added to the knowledge base, we will run the planning program on the test cases.

The process which we would like to model is embodied in the situation where a scientist, after hearing of a new laboratory technique, recognizes a laboratory problem for which the technique could be profitably applied. Thus, the first step of the process is to select a problem (from the casebook) after being presented with a new technique. Then the existing planning methods in the system would be applied on the problem again with the new laboratory technique encoded in the knowledge base. The next phase is deciding whether a new solution using the new technique offers any advantage over previous solutions. This would make use of the work described in Section 2.2 for abstracting and evaluating plans.

In summary, this process will make use of other developments in the MOLGEN project for applying newly discovered techniques and for evaluating the plans produced. These methods will be applied to form a discovery process by augmenting them with a casebook of experiments and a selection process for picking experiments.

2.3 Understanding Experimental Discrepancies by Hypothesis Formation

Plans for molecular genetics experiments have something in common with computer programs, summer vacation trips, and almost any kind of plan devised -- they don't always work. Although published reports about experiments usually say little about the techniques which failed, our analysis of the actual development of experiments has shown us that debugging is an essential and integral part of successful laboratory experimentation [14].

Pioneering work in the debugging of computer programs [37] used a process of program development and correction using a knowledge base about bugs. Sussman has advocated creating a rough version of a program on a first pass followed by local corrections by a gallery of critics. In Sussman's work, the ignoring of detail during the first pass is a source of power for the approach. In experiment debugging, we find an additional reason for attending to details during a second pass -- the knowledge necessary for deciding which of the details are in difficulty is not available at the time the experiment is designed. For example, many assumptions about the input samples are made when the experiment is designed and the validity of these assumptions cannot practically be tested until the experiment is performed.

Section 2.3.1 describes several distinct sources of error in experiment design. Section 2.3.4 describes a number of research issues.

2.3.1 Sources of Bugs

The complexity of most laboratory techniques is such that there are many ways for an experiment to go awry. The following diagram will be used to categorize the sources of bugs.