The task of manually sequencing a protein is a complex one amenable to automated planning.

In addition to automation, the SeqER architecture provides a computational framework for the empirical testing of different reasoning strategies for experiment design suggested by historical, philosophical, and psychological studies of scientists. This work supplements work on theory formation and revision (Darden 91). Rule-based and case-based reasoning has been observed in think-aloud protocols from a human expert planning protein sequencing experiments.

The following section describes the task of experiment planning. Section 2 describes the SeqER architecture and the KB component being used for protein sequencing experiment planning. Section 3 describes the analogical reasoning mechanisms of the Planner component. Section 4 presents an example of SeqER's operation for a protein sequencing experiment. Section 5 discusses related work, and Section 6 outlines some directions for future research.

1.1 Experiment Planning

Experiment planning is a complex task. The experiment should efficiently and inexpensively achieve its goals (e.g., test a hypothesis, determine a property, etc.). The use of resources (e.g., money, time, people, equipment, supplies) should be minimized. Planning may be complicated due to the unavailability of strong domain knowledge, established methodologies, or experienced personnel. In addition, uncertainty about experimental conditions, instrument readings, and the objects being manipulated must be dealt with. The results of executing a laboratory procedure may not be completely predictable in cases where domain knowledge is lacking or uncertainty exists. Thus, planning may need to be interleaved with execution such that the results of previous procedures help to determine the next procedure to execute.

This interleaving is necessary in experiments to sequence an unknown protein (polypeptide). The sequence of a protein (often hundreds of amino acids long), along with its shape and other properties, determines the structure (and hence the function) of the protein. There are 20^{200} possible sequences for a polypeptide of average length. SeqER plans an experiment to determine a polypeptide's amino acid sequence, N-terminal and C-terminal end groups (i.e., Formyl, Acetyl, Pyroglutamyl or Amide blocked, or unblocked), circularity, and disulfide bond locations (if any).

Various laboratory procedures can be performed to achieve these goals including acid and base hydrolysis, chemical and enzymatic cleavage, oxidation, Edman degradation, end group analysis, exopeptidase degradation, etc. The execution of many of these procedures is realistically simulated by the SequenceIt program(Place and Schmidt 92)[3], a tutorial program for protein sequencing techniques which we are using to test SeqER's plans in lieu of an actual laboratory. SequenceIt provides 35 laboratory procedures to choose from, each of which has several parameters that can be varied.

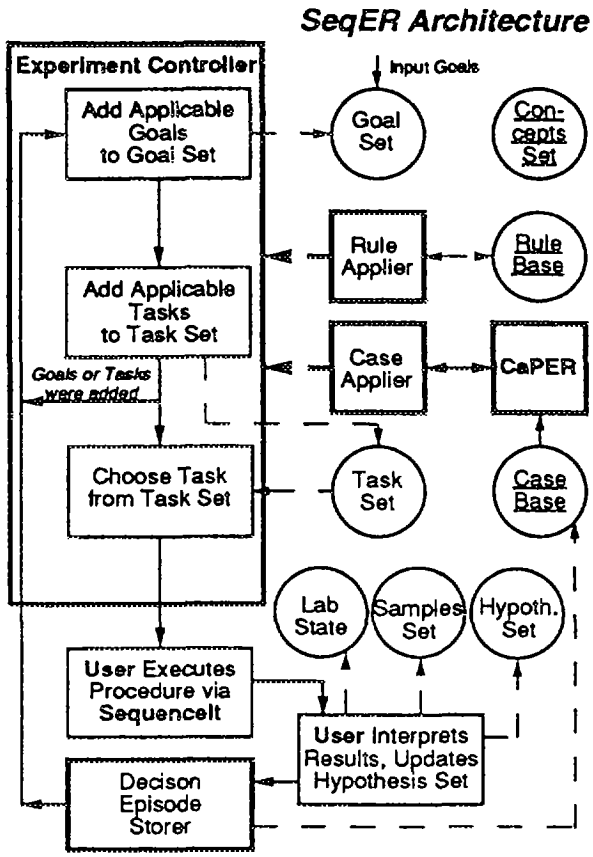

Figure 1: The SeqER Architecture: Main Modules and Process Flow

2. An Overview of SeqER

SeqER's top-level processes and modules are shown in Figure 1. The domain-independent Planner component consists of the Experiment Controller, Rule Applier, Case Applier, and Decision Episode Storer modules (shown as dark-outlined boxes). These last two modules, in conjunction with modules of the CaPER case-based planner, contain the mechanisms for reasoning by analogy (described in Section 3). The Rule Applier module matches and fires rules. The Experiment Controller module runs the top-level process (described in Section 2.2). (Solid arrows indicate flow of control. Dashed arrows indicate flow of information). The domain-specific KB component is described in the following section.

[3]This program was developed as part of the BioQUEST biology software project.

2.1 SeqER's Knowledge Base

SeqER uses a variety of kinds of knowledge to determine the subgoals of the experiment, the tasks that achieve them, and the task to be executed next. This knowledge is contained in the KB component, which contains both working knowledge[4] (knowledge about the problem at hand) and persistent knowledge (which persists between problems). (The latter is indicated in Figure 1 by circles with underlined labels; the former by circles with plain labels). Most of the KB is stored in a single large semantic network, implemented using the PARKA massively parallel, frame-based knowledge representation system (Spector et al. 90; Evett et al. 93). The following paragraphs describe the modules ("knowledge sources") within the KB component and give examples from the KB used for protein sequencing.

The **Concepts Set** consists of those frames in the KB representing domain concepts including domain objects (e.g., "Polypeptide", "Amino Acid", "Disulfide Bond"), properties (e.g., "Amino Acid Composition", "Sequence Length"), and procedures (e.g., "Acid Hydrolysis", "Clostripain Cleavage"). Procedures are considered (atomic) actions from SeqER's viewpoint. Actions can have roles (e.g., "Agent", "Object"), preconditions, filter conditions, parameters, and effects. A plan is a sequence of primitive actions, those that can be executed directly. Concepts are organized in an is-a (a-kind-of) abstraction hierarchy that supports inheritance. There are currently 450 concept frames in the protein sequencing KB.

Concepts may have instances in the world. For example, "Sample 101"[5], an instance of "Polypeptide Sample", could be a particular sample with particular properties from a particular experiment. Like "thing" concepts, action concepts may also have instances – termed "episodic actions (e-actions)". An e-action represents a particular potential occurrence of a primitive action involving particular objects, parameters, etc. For example, "Acid Hydrolysis 202", an instance of "Acid Hydrolysis", could be acid hydrolysis on Sample 101 with parameters duration = 24 hours and sample amount = 1 nMole. An e-action is a kind of episode. Other kinds of episodes include experiment episodes and decision episodes, both of which are part of the **Casebase** (see Section 3).

The **Goal Set** contains the current outstanding goals (subgoals) for the experiment such as "Amino acid composition for Sample 101 determined", "Disulfide bonds of Sample 202 broken", "Have overlapping sequenced fragments for Sample 303", etc. A goal may have an associated parent goal, sibling goals, task for which the goal is a precondition, and task(s) which achieves the goal. Goals are currently added to the Goal Set by the firing of production (if-then) rules that match structures in the KB or by their instantiation as preconditions of an action to be executed. The use of analogy to make use of previous subgoaling information is also being investigated.

The **Task Set** contains tasks currently being considered for execution. Tasks are e-actions and are either laboratory procedures (such as Aminopeptidase or Edman Degradation) or data procedures (such as Estimate Amino Acid Amounts or Assemble Sequence Fragments). A task may have an associated purpose (i.e., the goal it achieves) and preconditions. Tasks may be partially ordered. Tasks are currently added to the Task Set by the firing of production rules that match structures in the KB, especially goals in the Goal Set. The use of analogy to add tasks to the Task Set by reusing previous task-goal associations is being investigated.

The **Samples Set** consists of all instance frames representing samples in the current experiment. A polypeptide sample has an associated quantity, origin (the sample and procedure it was derived from), and properties (composition, sequence, length, etc.). SeqER starts with an initial sample of a polypeptide. Additional samples can be derived from this sample using cleavage procedures, performic acid oxidation, etc.

The **Lab State**, a module of the KB that is separate from the main semantic network, is a set of facts that represent the current state of the lab (the state of objects, equipment, etc.). This includes facts stating which data has been collected, which samples have been modified, etc. The Lab State and Samples Set are updated to reflect the effects of an experiment procedure's execution.

The **Hypothesis Set** contains frames representing hypotheses about the various properties (amino acid composition and sequence, disulfide bond locations, end groups, and circularity) of current polypeptide samples. An amino acid composition hypothesis is of the form "There are N A amino acids in Sample S" where N is a number, A is a kind of amino acid (e.g., Alanine, Arginine, etc.), and S is a polypeptide sample. A sequence hypothesis is of the form "There is a A amino acid in Position M in S", where M is a position in S's sequence. A disulfide bond hypothesis is of the form "There is a Disulfide Bond between Positions M_1 and M_2 in S" where M_1 and M_2 are the hypothesized locations of two Cysteines. Hypotheses may also be negative: i.e., "Sample S does not contain Serine". A hypothesis may have an associated confidence value (e.g., "high", "medium high", etc.), supporting evidence (procedures executed), and related hypotheses. Hypotheses are entered by the user based on his/her interpretation of the results of a lab or data procedure that was executed via SequenceIt.

The **Rule Base** contains production rules of three types: goal determination rules, task determination rules, and task selection rules. Rule antecedents are

[4] A.k.a. "working memory"

[5] A name consisting of a concept name and a numerical suffix designates an instance frame.

matched against the KB (semantic net and Lab State) to determine if a rule can fire. The consequents of fired rules can modify the working knowledge in the KB. Goal determination rules and task determination rules add goals to the Goal Set and tasks to the Task Set, respectively. Task selection rules are used to select among alternative, executable tasks (see Sections 3.3 and 3.4).

Some rules are domain-specific and were derived from analysis of human protocols and from domain textbooks, papers, etc. An example is the goal determination rule GDS1: "If the goal is 'Determine Polypeptide Properties for Sample S', then instantiate subgoals 'Determine Sequence for S', 'Determine Circularity for S',..."[6] Others are more generic rules such as the task determination rule TD1: "If there is a Goal G for Sample S and there is a procedure A which achieves G, then instantiate the task A for S."

2.2 SeqER's Top-Level Process

Given the overall goal(s) of the experiment and information about the initial situation, SeqER produces a plan. Planning and execution are interleaved in an experiment: after each procedure is selected by SeqER, it is "executed" by the user who then interprets the results of the procedure and supplies SeqER with new hypotheses to investigate.

For protein sequencing experiments, the input goal is to determine the amino acid sequence and other properties of a polypeptide. As initial input, SeqER is told of the quantity (and any other known properties) of a polypeptide sample whose properties are to be determined (via SequenceIt). It is also told of any unusual situations such as any lab equipment that is not functioning or other constraints (time limits, etc.) on the plan to be produced.

SeqER repeats the following process until the experiment successfully concludes (the polypeptide's properties have been determined) or fails (SeqER is out of sample or other resources, or it cannot find executable tasks for its current goals).

1. The Experiment Controller (EC) adds applicable goals (subgoals) to the Goal Set through the firing of goal determination rules.[7]
2. The EC checks each goal in the Goal Set to see if it is already established (by examining the KB). If the Goal Set is empty or all goals have been established, SeqER ends the experiment with success.
3. The EC adds applicable tasks (i.e., those achieving unestablished goals in the Goal Set) to the Task Set through the firing of task determination rules.
4. If new goals or tasks were added in the above steps, Steps 1-3 are repeated (because the new goals and tasks may suggest additional goals and tasks).
5. If no executable task can be found in the Task Set (and the Goal Set is not empty), SeqER ends the experiment with failure. Otherwise, the EC chooses a task (laboratory or data manipulation/analysis procedure) from the Task Set to execute. Task selection is done using previous decision episodes (see Section 3).
6. The task is presented to the user who executes it (via the SequenceIt simulator).
7. The user interprets the results of the procedure's execution and enters changes to the Lab State and Hypothesis Set. (Other changes to the Lab State are made automatically using knowledge of the procedure's default effects).
8. SeqER stores a new decision episode in the casebase which describes the task selection just made.
9. If the user judges the experiment complete, SeqER stores a new experiment episode and exits.[8] Otherwise, the process is repeated from Step 1.

3. SeqER's Use of Analogy

SeqER uses derivational analogy (e.g., (Carbonell 86)) to reuse previous planning experience. Derivational analogy differs from transformational analogy in that the latter makes use of the solution of the previous problem (e.g., a previously generated plan) whereas the former makes use of the reasoning used to produce that solution (e.g., the decisions made during the planning). Both kinds of analogy have been used by A.I. planning systems. Case-based planners such as CaPER typically use transformational analogy. Derivational analogy is used for planning by the PRODIGY/Analogy system where it provides a considerable reduction in search effort (Veloso 92).

SeqER captures planning experience in the form of decision episodes and experiment episodes. Both of these kinds of episodes are stored as cases in the casebase. An experiment episode describes an experiment that was performed and consists of a sequence of decision episodes. A decision episode (DE) describes the choice of a task (laboratory or data procedure) to execute. A DE consists of the set of tasks chosen from, the task chosen, and the reason(s) for the choice made. DEs capture control knowledge which can be reused in analogous situations. For example, suppose a decision was made in a particular situation to choose task A from the set of tasks $\{A, B\}$, where both tasks achieve the goal G. When faced with the same (or a similar)

[6] Where S is a variable whose value is an instance of Polypeptide Sample.

[7] Currently all firable rules are fired: no conflict set resolution is performed.

[8] Currently the user decides when the sequence has been accurately determined. Determining when hypotheses have been proven is a nontrivial task. SeqER might be modified to assist in this, perhaps by checking to see if the hypothesis set has quiesced.

choice in a similar future situation, a task analogous to A in that situation can be chosen, assuming that the choice of A was successful before.

In the event the original experiment failed some time after choice of A was made, however, it may be difficult to determine if the choice of A over B was successful or not. Determining which choice(s) led to failure is an instance of the difficult credit-blame assignment problem. When available, information about success or failure of a choice is also stored in the DE. SeqER currently assumes that a previous choice was successful unless it is obviously at fault. For example, the choice of A can be deemed a failure if A clearly did not achieve goal G when A was predicted to achieve G. Thus, in the future when the objective is to achieve G in similar circumstances, one has reason to avoid choosing A. Thus one can avoid possibly repeating one's previous mistake by pruning A from the Task Set.

The savings from reusing control information is greatest when there is a large number of alternatives to consider and through knowledge of previous choices a more informed choice can be made from among them, rather than pursue the path (the sequence of tasks) from each. Pursuing a path can be expensive in SeqER because planning and execution are interleaved and there is no ability to backtrack by undoing choices made. Knowledge of which alternative previously led to success can be used to select that alternative (or an analogous one) in the current situation. Knowledge of which alternatives previously led to failure can be used to prune those alternatives (or analogous ones) from the Task Set. The danger in pruning alternatives is the possibility of not finding a more optimal path to a solution.

When SeqER must decide which task in the Task Set to execute next, it searches for episodes in which similar decisions were made. If it finds such a DE, it can opt to choose the task in the current ("target") situation that is analogous to the task chosen in the previous ("source") episode. An alternative approach to using DEs to guide task selection, or making an arbitrary choice, is the use of heuristic task selection rules (see section 3.4).

3.1 Acquiring Decision Episodes

Initial DEs were obtained by analyzing experiment logs kept by a human expert user of the SequenceIt program. Thus some of the user's expertise is captured in these DEs and can be use of by SeqER. These initial experiments are "correct" – the choices made led eventually to a plan that determined a polypeptide's properties. These experiments, however, are not assumed to be optimal – it may be possible to find a successful experiment that has fewer procedures or that consumes fewer resources.

Each time SeqER chooses among tasks, a new DE is created to describe this choice and is stored in the casebase. In this manner, SeqER can improve over time as it accumulates additional planning experience. It is expected that SeqER's knowledge base will grow quite large as these DEs accumulate.

3.2 Decision Episode Retrieval

When faced with choosing a task from the Task Set, SeqER attempts to retrieve similar decision episodes from its casebase. SeqER's Case Applier component retrieves DEs using the case retrieval mechanism of the CaPER case-based planning system. Through the use of massive parallelism, CaPER can retrieve cases efficiently from a very large, unindexed memory (Kettler et al. 93). Because this case memory is unindexed, there is high flexibility as to which features of the current situation (i.e., Lab State, Goal Set, Hypothesis Set, current samples, etc.) can be used as part of the retrieval probe.

For protein sequencing, SeqER currently uses a simple probe consisting of the type and object of each executable task in the Task Set. As a simple example, consider the case where the current task set contains two target tasks: (TT_1) Acid Hydrolysis on Sample S1 and (TT_2) Base Hydrolysis on S1. A retrieval probe will be issued to find all DEs where the alternatives included Acid Hydrolysis on Sample S and Base Hydrolysis on Sample S (where S is any previous sample). If a single source DE is returned, it is selected for application (as described in the next section). If no source DE is returned, SeqER must use other means (perhaps even guessing) to choose among the target tasks. CaPER's retrieval flexibility makes it easy to add other features, such as the hypotheses about a sample, to the retrieval probe.

In addition, SeqER can make use of CaPER's probe relaxation mechanism in the event that no cases match a retrieval probe (or some feature in the retrieval probe). Features in the retrieval probe can be relaxed using information from the concept is-a hierarchy. For example, SeqER may have to choose between the tasks Clostripain Cleavage and Acid Hydrolysis. It issues a probe containing these tasks and gets no matching cases. Using its taxonomic knowledge that Clostripain Cleavage is-a Enzymatic Cleavage Procedure and that Acid Hydrolysis is-a Hydrolysis Procedure, SeqER could issue a new, more general probe, replacing "Acid Hydrolysis" with "Hydrolysis" and/or replacing "Clostripain Cleavage" with "Enzymatic Cleavage".

SeqER uses a simple similarity metric to rank the DEs retrieved using the probe. Each of these DEs contains one or more source tasks that match some target task in the retrieval probe. SeqER counts the number of tasks that each source DE has in common with the target DE. This score ranges from 1 to K, the number of executable tasks in the Task Set. In the above example, this score could be 1 or 2. DEs with scores below 2 are rejected.[9] Multiple DEs with the same similar-

[9]This heuristic eliminates DEs having only a single task

ity score are ordered such that those DEs that have the fewest extraneous tasks (tasks with no analogues among the target tasks) are preferred. The adequacy of this simple similarity metric needs to be determined through further empirical study.

SeqER's Case Applier next attempts to find a complete mapping between each source DE retrieved and the target situation. A one-to-one mapping must be found between target entities – target tasks (and their samples) matched during case retrieval – with source entities – tasks (and samples) in the source DE. It is possible that several such mappings may exist between a source DE and the target situation. SeqER currently considers only the first such mapping it finds. If no such mapping can be found for a source DE, it is rejected.

SeqER next checks each source DE for which it has established a mapping(s) to determine if a target task was found that is analogous to the task chosen in the source DE. If no analogous target task exists, the source DE is rejected. For example, consider a source DE that has two tasks, A_S and B_S, which match two target tasks A_T and B_T, respectively. Suppose the source DE has an additional task C_S which was the task chosen but has no analogue among the target tasks. This source DE will be thus be rejected since there is no choice among the targets to make that is analogous to C_S.[10]

3.3 Decision Episode Application

SeqER considers each source DE returned by the retrieval/mapping procedure described above. Source DE's are considered in decreasing order of their similarity scores. For each source DE, SeqER checks the reason(s) for the choice that was made. Reasons for a choice are: the choice is supported by a rule(s), the choice is supported by a (analogous) DE(s), or both. It is also possible a choice was made arbitrarily. SeqER looks for reasons analogous to those from the source DE that can justify choosing the analogous target task in the current situation.[11] In addition, SeqER also considers any information it has as to the success or failure of choice made in the source DE. This information can also be used to justify a particular choice among the target tasks.

To continue the previous section's simple example, suppose SeqER is considering a source DE – DE3 – where a choice was made between source tasks (ST_1) Acid Hydrolysis on sample S2 (previously chosen) and (ST_2) Base Hydrolysis on S2. The mapping {(TT_1 :: ST_1), (TT_2 :: ST_2), (S1::S2)} is found.[12] Since ST_1 was previously chosen, SeqER considers choosing the analogous target task, TT_1 (Acid Hydrolysis on S1).

Suppose the reason for DE3 was the applicable task selection rule "Rule 11: If $task_A$ has fewer harmful side effects than $task_B$, choose $task_A$ over $task_B$" with variable bindings $task_A = ST_1$, $task_B = ST_2$. The Rule Applier checks to see if the rule applies in the current situation when instantiated with the proper substitutions from the mapping (TT_1 for $task_A$, TT_2 for $task_B$). If so, Rule 11 justifies choosing TT_1 over TT_2. Although it is possible that Rule 11 could have been checked for application without first retrieving any DE, there are advantages in the approach used (see Section 3.4).

If, however, the reason for choosing ST_1 in DE3 had been another decision episode (say DE2), then SeqER assumes that DE2, by virtue of its justifying the choice of ST_1 in DE3, also justifies choosing TT_1 in the current situation. In other words, because the current situation is analogous to DE3 which is analogous to DE2, the current situation is analogous to DE2 by transitivity. Thus, DE3 would justify choosing TT_1.

Finally, suppose ST_1 was chosen arbitrarily in DE3. In this event, if it is known that choosing ST_1 (rather than ST_2) proved to be a "good" decision, SeqER is justified in choosing TT_1 given the assumption that DE3 is similar to the current situation. If the quality of choice of ST_1 is not known, however, then SeqER will, in the absence of any other justification, consider DE3 sufficient justification for choosing TT_1.[13] This is an example of using analogy to make a choice when no stronger domain knowledge such as a rule is available, rather than just make an arbitrary choice.

For sources DEs with the same similarity score, SeqER prefers the source DE that suggests (by analogy) the choice of a particular target task for which the best justification exists in the current situation. Justification by a rule is preferable to justification by a case. Both are preferable to choosing a target task analogous to one chosen arbitrarily in some source DE. The target task, along with the source DE that suggested it, is returned to the Experiment Controller for execution by the user.

The justification mechanism in SeqER is being fine-tuned through empirical evaluation. There are several possible extensions including the use of multiple source DE's to justify the choice of a particular target task. Multiple source DEs might suggest the same target

analogous to a target task. This is not considered a sufficient basis for an analogy between the source DE and the target situation.

[10]This information could be of use in pointing out tasks that perhaps should be under consideration but are not. If the source DE describes a situation sufficiently similar to the current situation, the system might decide to look for and instantiate (via a rule, etc.) a task analogous to C_S and add it to the Task Set.

[11]The importing of reasons from the source DE to the target situation is a kind of transformational analogy.

[12]Note that TT_i and ST_j are constants standing for particular tasks. S1 and S2 are actual sample names.

[13]This approach is more warranted when there is reason to believe the source DE describes a good (or at least a "non-bad" choice), such as when the source DE was from an experiment performed by a human expert.

task and hence provide better justification for choosing it than a single source DE could.

3.4 Cases and Rules

The interaction of rules and cases is being explored in SeqER. Analysis of human expert protocols in this domain indicates that both are used to determine goals/tasks and for task selection. These types of rules are thus used by SeqER. The integration of rule-based reasoning and case-based reasoning has proven useful for systems in other domains (e.g., (Rissland and Skalak 91)).

Our initial approach in SeqER is to minimize the use of task selection rules. Often such rules are not available, particularly in less understood domains, or are highly specific as to their applicability. Our aim is to see how far SeqER can get by its use of analogy.

In SeqER, cases can be used to organize the task-selection rules. Consider the example from the previous section in which the justification for DE3 is Rule 11. When DE3 is retrieved, Rule 11 is "retrieved" and checked to see if it applies in the current situation. If so, Rule 11 justifies choosing TT_1 . An alternate approach would be to consider all such rules at task selection time and see if any can apply, before we retrieve any DEs. Rule 11 would thus be found and applied directly, without retrieving DE3. Although this approach is practical for small rulebases, it becomes less practical to always have to check each rule in a large rulebase for applicability at every task selection step because rule matching can be expensive. With SeqER's approach, a task-selection rule is not considered until it is suggested by a case (such as Rule 11 was suggested by DE3). Some recent psychological results have shown how some experts use cases to organize rules (Boshuizen and Schmidt 92).

Cases can also provide a context in which a task selection rule applies. This is especially useful for heuristic rules that only work in certain situations. For example, suppose that we have task selection Rule 22 that only works in very limited situations and that was applied with successful results in DE5. If we have retrieved DE5 and are considering applying Rule 22 to the target situation, we can be more confident that Rule 22 will work since our current situation is similar to DE5.

One extension to SeqER would be to add a mechanism to induce task selection rules from similar cases. Another extension would be to induce the context of applicability for task selection rules from similar cases where the rule was applied with success.

4. An Example of SeqER's Operation

The following text describes part of an actual SeqER run. SeqER was initially given 200 nMoles of Beta-Endorphin protein to determine its sequence and other properties. The casebase used here contained 12 experiments of fairly short length (15-20 DEs each – a total of 207 decision episodes). All of these experiments were encoded from the logs of a human domain expert. The total KB size (conceptual and episodic memory) was over 1400 frames. To test the ability of reasoning by analogy, task selection rules were disabled and cases alone used to choose among tasks.

SeqER first instantiated the initial goal (Goal-1325: PP-Props-Determined for Sample-1324) for the new experiment, Experiment-1323. It then fired as many goal and task determination rules as possible, until no new rules could be fired. SeqER now had to choose among the following executable target tasks (procedures/e-actions) from the Task Set: Base-Hydrolysis-1424 of Sample-1324, Acid-Hydrolysis-1426 of Sample-1324, and Performic-Acid-Oxidation-1430 of Sample-1324.

Three candidate source decision episodes were retrieved (DE-1890, DE-1844, and DE-1793), each with a score of 2 (they each have two source tasks matching the above target tasks). Each decision episode was considered and all were found to have the same reasons: NIL (i.e., no justification was given in the source DEs). In the absence of source DEs with justifications, SeqER chose one of these (DE-1890, from an experiment to sequence Bombesin) where the choice of source task Acid-Hydrolysis-1888 on Sample-1886 (over Base-Hydrolysis-1889 on Sample-1886) was successful. Using the mapping between these source tasks and the target tasks, the analogous target task to be chosen (Acid-Hydrolysis-1426 of Sample-1324) was determined.

The user executed Acid Hydrolysis (24 hours duration, 1 nMole of Sample-1324) using SequenceIt which successfully achieved the goal of obtaining data on the amino acid composition of Sample-1324. This goal was marked achieved (and removed from the Goal Set), and task Estimate-Composition-1370, for which this goal was a precondition, was added to the Task Set. (Base-Hydrolysis-1424 was no longer needed to achieve this goal and was thus removed from the Task Set.) A new DE was created to record the choice of Acid-Hydrolysis-1426 and DE-1890 and was stored as the reason justifying this choice.

Other rules were fired, and eventually SeqER next chose between Estimate-Composition-1370 and Performic-Acid-Oxidation-1430, both for Sample-1324. Three candidate source decision episodes were retrieved but each only had a single source task analogous to a target task (a score of 1). This was not sufficient for an analogy, and the source DEs were rejected. SeqER arbitrarily chose Estimate-Composition-1370.

The user used the acid hydrolysis data in SequenceIt (obtained in the previous DE) to successfully estimate the amino acid amounts in Sample-1324. The user returned these estimates as 20 medium-high confidence hypotheses: 2 Alanines, 2 Aspartic Acids, 5 Lysines, etc. These were added to the Hypothesis Set, and a new DE was created to describe the choice and results

of this action (reason was "Arbitrary Choice"). The target goal of obtaining amino acid composition hypotheses for Sample-1324 was marked achieved.

The addition of new hypotheses triggered (via the firing of rules matching the Hypothesis Set and Concepts Set) the addition to the Task Set of laboratory procedures that cleaved on the hypothesized amino acids in Sample-1324. SeqER next chose Performic-Acid-Oxidation-1430 over the following tasks (all for Sample-1324): Estimate-Length-1356, Thermolysin-Cleavage-1476, Lysobacter-Protease-Cleavage-1478, and Trypsin-Cleavage-1480. Performic Acid Oxidation was executed to derive a new sample (instantiated as Sample-1489) from 100 nMoles of Sample-1324. SeqER then continued to perform additional procedures on this new sample, which lacked the disulfide bonds that can complicate sequence determination.

This example is illustrative of several aspects of SeqER's operation in this domain and in general. The features used in the retrieval probe and in the similarity metric are a kind of implicit domain knowledge. In the protein sequencing domain the use of only the target tasks (and samples) in case retrieval/selection often results in a source DE being discarded due to it only matching a single target task. This mainly occurs when the source DE was from a human-planned experiment, where the alternatives being considered at each decision point are few. We conjecture that the human expert is using additional domain knowledge to limit the alternatives under consideration. SeqER, in contrast, adds all the tasks it can (via the firing of rules) to the Task Set and then chooses among them. SeqER's approach results in a systematic enumeration of the alternative tasks but these include ones that might not be particularly useful to execute. To mimic the human expert's strategy, additional domain knowledge could be incorporated into the goal and task determination rules.

In addition, it has been observed that SeqER often lacks the ability to make fine discriminations among source DEs that have highly similar source tasks. When selecting which of these DEs is most similar to the target situation, it would be useful to compare the (hypothesized) known properties of samples in their source tasks with those samples in the target tasks. This is currently being investigated. The modular nature of the KB makes it easy to add, delete, or modify its contents based on this kind of empirical evaluation of the system's operation.

The size of the KB can be quite large. Experiments to sequence small polypeptides (20 or so amino acids) often have more than 20 DEs. Some large polypeptides (hundreds of amino acids) sequenced by a human expert have over a hundred DEs. SeqER must be able to support these large casebases of thousands of frames. The parallel case retrieval mechanisms of CaPER/Parka that are used have been proven efficient on large casebases (tens of thousands of frames) in other domains and should scale well for SeqER casebases too.

5. Related Work

The MOLGEN (Stefik 81) system used generative planning techniques along with constraint posting and meta-planning to design experiments for gene cloning. Unlike MOLGEN, SeqER must interleave plan generation and execution. SeqER also uses different planning techniques, such as case-based planning, than MOLGEN did. More recent work using MOLGEN for automated discovery is described in (Friedland and Kedes 85).

The use of derivational analogy to control a generative planner has been investigated in the PRODIGY/Analogy system (Veloso 92). SeqER also uses analogy for control of a (simpler) planner but employs different case representations and retrieval techniques and interleaves planning with execution. Other systems have used derivational analogy for planning in domains other than experiment planning including the APU system (Bhansali and Harandi 93) for the synthesis of UNIX programs and various systems that use derivational analogy to design objects (see (Mostow 90) for an overview of these).

Several systems have used analogy for scientific reasoning including the PHINEAS system (Falkenhainer 90), which explained new scientific phenomena by analogy to previously-explained phenomena, and the PI system (Thagard 88). Also related to the general domain of SeqER is the HYPGENE/GENSIM system (Karp 89), which used iterative design of hypotheses to improve the predictive power of theories. There has also been some research in psychology on the task of experiment design (e.g., (Klahr et al. 90)).

6. Directions for Future Research and Conclusions

There are many possible interesting extensions to the protein sequencing configuration of SeqER and the architecture in general to be investigated after further empirical study of the prototype. The hypothesis set could be monitored to collect data about how it changes, based on the results of executing a lab procedure, and checked for weak or inconsistent hypotheses, which could be used to suggest appropriate lab procedures to remedy these deficiencies. SeqER could focus more on plan optimality (i.e., minimize resource consumption, etc.). The capturing of more detailed information on quality of choices allows better evaluation of whether an analogous choice should be made in a future situation. The reuse of larger units (i.e., several consecutive actions) from previous plans where possible is also being investigated.

SeqER's architecture has allowed us to explore the application of derivational analogy and other planning

techniques to scientific experiment planning. Relevant planning experience can be efficiently retrieved and applied to select tasks for execution in protein sequencing experiments. New experience is encoded as new cases in the casebase. Knowledge about laboratory and data manipulation/analysis procedures, laboratory objects, goals, methodology, hypotheses, and previous plans is represented as frames, rules, and cases in a modular knowledge base component. This KB component captures domain-specific knowledge while the reasoning mechanisms of the Planner component are domain-independent. Thus it is anticipated that SeqER could easily be configured for other experiment planning domains.

We are currently investigating potential domains where there is a real need for automation to complement the skills of human scientists. SeqER, acting as an assistant, could accurately recall and present relevant planning experiences from the past, advise on which procedure to select next, and maintain up-to-date information about current hypotheses.

Acknowledgments: We wish to thank Dr. James Hendler for his direction and support of the CaPER project; Dr. Joshua Lederberg and his associates for feedback on SeqER and experiment planning; Dr. Allen Place for his making the SequenceIt program available to us; Ting-Hsien Wang and especially Nathan Felix for their knowledge acquisition work for SeqER; and Bill Andersen for his development of new Parka code and tools. The CaPER part of this research was supported in part by grants from NSF (IRI-8907890), ONR (N00014-J-91-1451), and AFOSR (F49620-93-1-0065).

References

Bhansali, S., and Harandi, M.T. 1993. Synthesis of UNIX Programs Using Derivational Analogy. *Machine Learning*, Vol. 10, pp. 7-55.

Boshuizen, H.P.A., and Schmidt, H.G. 1992. On the Role of Biomedical Knowledge in Clinical Reasoning by Experts, Intermediates, and Novices. *Cognitive Science*, 16, pp. 153-184.

Carbonell, J.G. 1986. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In *Machine Learning*, Vol. 2, Eds. R.S. Michalski, J.G. Carbonell, T.M. Mitchell. San Mateo, California: Morgan Kaufmann Publishers.

Darden, L. 1991. *Theory Change in Science: Strategies from Mendelian Genetics.* New York: Oxford University Press.

Evett, M.P.; Hendler, J.A.; and Spector, L. 1993. Parallel Knowledge Representation on the Connection Machine. *Journal of Parallel and Distributed Computing*, forthcoming.

Falkenhainer, B. 1990. A Unified Approach to Explanation and Theory Formation. In *Computational Models of Scientific Discovery and Theory Formation.* Eds. J. Shrager and P. Langley. San Mateo, California: Morgan Kaufmann Publishers, pp. 157-196.

Friedland, P., and Kedes, L.H. 1985. Discovering the Secrets of DNA. *Communications of the ACM*, Vol. 28. No. 11., November 1985, pp. 1164-1186.

Karp, P.D. 1989. Hypothesis Formation and Qualitative Reasoning in Molecular Biology. Doctoral Dissertation Stan-CS-89-1263, Stanford University.

Kettler, B.P.; Hendler, J.A.; Andersen, W.A.; and Evett, M.P. 1993. Massively Parallel Support for Case-Based Planning. In *Proceedings of the Ninth Conference on Artificial Intelligence Applications* (IEEE). Held in Orlando, Florida, March 1-5, 1993. Washington, D.C.: IEEE Computer Society Press, pp. 3-9.

Klahr, D.; Dunbar, K.; and Fay, A.L. 1990. Designing Good Experiments to Test Bad Hypotheses. In *Computational Models of Scientific Discovery and Theory Formation.* Eds. J. Shrager and P. Langley. San Mateo, California: Morgan Kaufmann Publishers, pp. 355-402.

Mostow, J. 1990. Design by Derivational Analogy: Issues in the Automated Replay of Design Plans. In Carbonell, J.G., *Machine Learning: Paradigms and Methods.* Cambridge, Massachusetts: The MIT Press, pp. 119-184.

Place, A.R., and Schmidt, T.G. 1992. User's Guide for SequenceIt!: An Apprenticeship in the Art and Logic of Protein Sequencing. Center of Marine Biotechnology, Maryland Biotechnology Institute and Project BioQUEST.

Rissland, E.L., and Skalak, D.B. 1991. CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, Vol. 34, pp. 839-887.

Spector, L.; Hendler, J.A.; and Evett, M.P. 1990. Knowledge Representation in PARKA. Technical Report 2410, Department of Computer Science, University of Maryland at College Park.

Stefik, M. 1981. Planning with Constraints (MOLGEN: Part 1). *Artificial Intelligence*, Vol. 16, pp. 111-140.

Thagard, P.R. 1988. *Computational Philosophy of Science.* Cambridge, Massachusetts: The MIT Press.

Veloso, M.M. 1992. Learning By Analogical Reasoning in General Problem Solving. Doctoral Dissertation CMU-CS-92-174, School of Computer Science, Carnegie Mellon University.