Partial Compilation of Strategic Knowledge[1]

Russ B. Altman and **Bruce G. Buchanan**
Knowledge Systems Laboratory
Stanford University

Abstract

Many system building efforts in artificial intelligence intentionally begin with expressively rich and flexible declarative structures for the control of problem solving–especially when the best problem solving strategies are not known. However, as experience with a system increases, it sometimes becomes desirable to compile declarative knowledge into procedures for purposes of efficiency. We present a paradigm for compilation which begins with declarative *opportunism*, moves to a phase of heuristic implementation of a *partial plan* and finally evolves into a fully elaborated procedure. We use the PROTEAN geometric constraint satisfaction system as an example. Using results from a purely declarative structure, we were able to compile strategic knowledge into a procedure for planning a solution. The problem solving behavior of the new system is reported.

I. Introduction

Knowledge compilation offers an engineering solution to the problem of combining the flexibility of a declarative representation of knowledge with the efficiency of a more procedural representation. For applications in which knowledge is changing frequently, the benefits of declarative representations may outweigh considerations of efficiency (especially during development). For others, in which run-time efficiency is more important, the use of declarative representations for initial development must be followed by *compilation* of knowledge. As knowledge-based systems become larger, there is increasing use of separate meta-level knowledge structures (sometimes called strategic or control knowledge) to reduce the complexity and increase the understandability of these systems [Davis, 1980, Clancey, 1985, Hayes-Roth, 1985, Hewitt, 1972, McDermott, 1978]. In this paper we show the results of compiling parts of this strategic knowledge, represented declaratively, into a partial plan that instantiates major control decisions.

It is useful to have a rational method by which the transition from declarative to procedural forms of strategic knowledge can be made gracefully. In this paper, we argue:

1. That the separation between strategic knowledge and domain knowledge (as in the PROTEAN/BB1 blackboard system) is useful in the development of efficient problem solving strategies. We suggest a three stage paradigm with which this development can usefully be viewed.
2. That if strategic knowledge is represented declaratively and separated from domain problem solving knowledge, then compilation of *strategic* knowledge can be performed and integrated within *domain* problem solving knowledge.
3. That the compilation of parts (but not all) of the problem solving knowledge yields plans in which flexibility is sacrificed for efficiency. These plans embody a set of decisions that may anticipate global problem solving strategy better than more locally focussed strategy knowledge.

We substantiate these claims with examples from the PROTEAN system for the determination of protein structure [Altman and Jardetzky, 1986, Brinkley *et al.*, 1986, Hayes-Roth *et al.*, 1986b]. PROTEAN is a geometric constraint satisfaction system described in section II. In one version of this system, Hayes-Roth and coworkers have shown that declarative control structures can be used to control reasoning about constraint satisfaction in spatial assembly problems [Hayes-Roth *et al.*, 1986a]. We have compiled elements of this strategic knowledge and have been able to implement plans to guide problem solving without any modification of the basic domain problem solving actions. The plans prescribe partial sequences of actions. Portions of the problem solving for which there is no plan prescription are controlled opportunistically with heuristics.

A. Three Phases of Development for Procedures

The development of a computational procedure for the solution of difficult problems can be usefully divided into three phases. The first phase, which we call the *opportunistic* phase, is characterized by the use of architectural frameworks in which there is considerable freedom for a designer to experiment with different formulations of the search and different strategies for controlling it.

Having gained experience from work within the opportunistic phase of development, we can enter the *partial plan* phase. A partial plan provides an incomplete specification of the actions required for a solution. In this phase a designer draws upon the heuristics and experience gained during experimentation to increase efficiency with a more rigid problem solving control plan. These plans are not fully prescriptive for problem solving, however, and some of the declarative strategic knowledge may remain when the plan prescribes nothing.

[1]This work was funded in part by the following contracts and grants: NIH GM07365, DARPA N00039-83-C-0136, DARPA N00039-86-C-0033, NIH RR-00785, NASA-Ames NCC-2-274, Boeing Computer Services W271799, and a gift from Lockheed Corp. We would like to thank Alan Garvey, Craig Cornelius and Barbara Hayes-Roth for discussion of their experimental results. We also thank Jim Brinkley, Bruce Duncan, John Brugge and Oleg Jardetzky for collaborative research on PROTEAN.

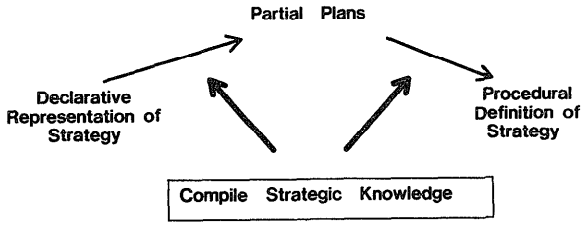

Figure 1: Three suggested stages in the development of strategies. The transition from declared strategies to more procedurally defined ones involves the compilation of strategic knowledge.

The plans provide more direction and purpose to the problem solving than strategies with a shorter horizon. Thus, a plan imposes a particular detailed set of steps on problem solving and so reduces the amount of computational effort spent on choosing among different choices.

Experimentation with a system in the partial plan stage may lead to refinement of the plan details. We believe, but have not shown, that compilation of strategy knowledge can be automated when the syntax of both the declarative strategy knowledge and the domain knowledge structures is known. When the plan becomes fully elaborated it can be called a heuristic procedure for solving the problem. The *procedure* phase requires little computational effort in choosing control alternatives–all decision points are predefined and the criteria for selection are predetermined. We have found that this computational saving comes at the cost of problem solving flexibility.

B. Compilation of Knowledge

In a changing, experimental setting it is useful to represent strategy heuristics with declarative data structures in order to provide an environment for experimentation with different strategies. However, when there is evidence that certain strategies are superior to others, it may become desirable to incorporate these strategies more directly into the solution for efficiency.

Compilation of declarative strategic knowledge is characterized by a move from a **description** of desirable actions to **prescriptions** for action. Systems which allow descriptive strategic statements are faced with the task of *interpreting* these statements and *matching* them with feasible actions in order to identify desirable actions [Hayes-Roth *et al.*, 1986a]. Davis [Davis, 1980] referred to the interpretation of such meta-level strategic statements as content-directed invocation. However, the cost of using generic control statement interpreters and action-matchers may not be warranted if there is a procedure which can identify the most desirable actions using specific domain knowledge. Such a procedure bypasses the use of all-purpose interpreters and matchers, and thereby improves efficiency.

We have manually constructed special-purpose, domain-dependent procedures for making strategic decisions in situations for which criteria have become clear from experimentation. These procedures eliminate the need for interpretation of control strategies and the overhead of matching these strategies with potential domain actions. They gain problem solving leverage by using knowledge of the application domain, and as such can be considered domain problem solving actions. The key step in compilation is creation of a partial plan, an abstract sketch of how to solve the whole problem which is stylized enough to allow straightforward translation into procedures. We do not assume that the plan is complete, however, and thus must abe able to solve problems with partially compiled, partially interpreted control strategies. Our procedures, therefore, produce a partial *plan* for the solution of the problem. Section II shows how the ideas apply to PROTEAN. Section III illustrates how this plan interacts with data driven control for a particular problem.

II. Geometric Constraint Satisfaction in PROTEAN

A. The PROTEAN System

PROTEAN is a system for determining the structure of protein molecules from experimental data. The system and motivations are described in detail in [Altman and Jardetzky, 1986, Brinkley *et al.*, 1986, Hayes-Roth *et al.*, 1986b], but are summarized for present purposes. PROTEAN begins with a number of abstract or elementary objects (atoms or groups of atoms with fixed physical relationships) and constraints among the objects. A constraint typically specifies a range of distances between two points.

PROTEAN makes "partial arrangements" of subsets of the objects in three dimensions, and then combines partial arrangements into a final solution space. In a single partial arrangement, a coordinate system is defined around a single object (called the **anchor**), and positions of other objects (**anchorees**) relative to the anchor are defined (see Figure 2). Except for the anchor, an object may have more than one "legal location" in which its positional constraints are satisfied. A "coherent instance" is a list of single locations for each object such that all constraints are satisfied. The set of all coherent instances represents the set of all structures of the protein that are consistent with the experimental constraints.

PROTEAN has a basic set of actions that have been procedurally defined within domain problem solving knowledge sources. They include:

- ANCHOR [B to A]: finds all locations for anchoree B relative to anchor A (in a partial arrangement) which are consistent with the constraints between A an B. Figure 2 shows the accessible volume of two anchorees relative to a fixed anchor (HELIX-5).
- YOKE [B and C with respect to A]: reduces the list of locations of anchorees (B and C) in the space of anchor A by pruning locations that are incompatible with the constraints between B and C. Figure 3 shows Helix-3 and Helix-1 after YOKING. Their accessible volumes have been reduced by consideration of the constraints between them (cf. Figure 2).
- APPEND [C to B with respect to A]: finds all locations of object C relative to an anchoree B in the space of anchor A. This involves finding all locations of C relative to B and B relative to A and then producing the cross product to get all locations of C relative to A. Figure 3 shows Helix-

2 positioned with an APPEND action by considering its constraints to Helix-3.

- CONSOLIDATE [objects with respect to A]: finds the set of locations (one from each object's accessible volume) that constitute a "coherent instance."

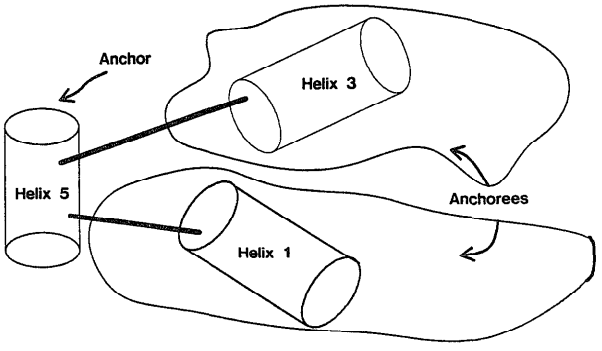

Figure 2: PROTEAN's basic problem solving action, ANCHOR. Legal locations are shown as accessible volumes around each anchoree.

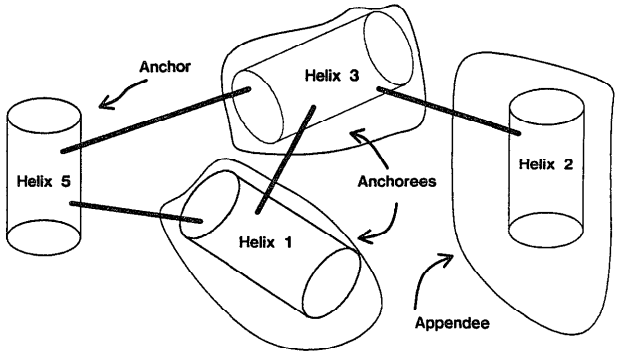

Figure 3: PROTEAN's basic problem solving actions, YOKE and APPEND.

A partial arrangement can be considered a constraint satisfaction network in which each node is an object with a list of locations and each arc between nodes represents constraints on the relationships between pairs of locations taken from the nodes. We have shown elsewhere [Brinkley *et al.*, 1986] that the anchor action corresponds to creating a constraint network that is node consistent in the terminology of Mackworth [Mackworth, 1977]. Yoking corresponds to checking for consistency of the arcs. Consolidation is equivalent to a backtrack search for solutions to the constraint network. Backtrack search is computationally prohibitive, and it can be made tractable by pruning the set of initial locations with anchor and yoke operations. PROTEAN's problem solving repertoire of four primitive actions has been implemented as a set of domain knowledge sources in the BB1 blackboard environment [Hayes-Roth, 1985]. Each action is represented as a domain Knowledge Source (KS) which is triggered when a relevant change is made to the problem solving ("domain") blackboard. A triggered KS is instantiated as a Knowledge Source Activation Record (KSAR) for each context in which the action becomes feasible, and is placed on the agenda. Domain Knowledge Sources are generally procedural statements of how to perform calculations and make appropriate changes on the problem solving blackboard. A separate control facility is then used to *rate* feasible actions and determine which actions should be performed.

B. Controlling PROTEAN's Reasoning

The problem of arranging objects in three dimensional space under constraints is known as bin packing, and is NP-complete. PROTEAN is able to solve such a combinatorially explosive problem partly because it can make reasoned choices about the best objects and best actions on which to focus at each stage of problem solving. The strategic choices that must be made by PROTEAN include:

1. How many partial arrangements should be created?
2. Which objects should be included in the partial arrangements?
3. Which objects should be designated the anchors of the partial arrangements?
4. Should an ANCHOR or APPEND actions be used to position a particular object within an arrangement?
5. In what order should YOKE actions be applied to most quickly reduce the size of the accessible volumes?
6. When are two partial arrangements ready to be merged together?
7. When should a partial arrangement be CONSOLIDATED (because pruning techiques have reaced a point of diminishing returns)?

In addition to our continued development of an explicit, declarative version of the strategy, there is a need for a systematic compilation of the method. Thus, part of our research focuses on the development of a straightforward procedural statement of how to make these choices, which in BB1 are called "control problems."

The key characteristic of this implementation of PROTEAN in BB1 is that there is a separation of the mechanism which generates feasible actions from that which selects actions for execution. When a control problem arises, the system can look to the agenda of feasible actions for a complete set of alternatives, and choose among them. The process of compilation of strategic decisions reduces the frequency at which the complete agenda must be examined.

C. Uncompiled and Compiled Control Knowledge in PROTEAN

The ACCORD language has been developed to define high level "control sentences" which declaratively and indirectly specify problem solving actions [Hayes-Roth, 1985]. For example, in choosing the best anchor for a partial arrangement, one control sentence reads:

> ORIENT a PARTIAL-ARRANGEMENT about a LONG, RIGID, CONSTRAINING SECONDARY-STRUCTURE.

This sentence is interpreted and matched with each task on the agenda (called a KSAR) in order to determine an overall rating for the task. The "action-type" of the KSAR is compared and scored relative to the action-type **ORIENT**, the potential anchor is checked and scored with respect to being a **SECONDARY-STRUCTURE** as well as being **LONG, RIGID** and **CONSTRAINING**. The definitions of these modifiers are stored in a knowledge base as quantitative rating functions. The KSAR action that best matches this declaration is chosen for execution.

This control mechanism is flexible since it can handle a wide variety of problems, and be used for explaining its selection [Schulman and Hayes-Roth, 1987]. It also allows different modifiers to be easily tested and results to be compared [Garvey *et al.*, 1987]. In short, it is convenient for experimenting to find specific rules for solving classes of problems. It is expensive, however, since each potential anchor must be rated with respect to a number of different modifiers. In addition, it takes a best-first approach to control, and assumes that optimal decisions locally will produce good global performance. Each control sentence is meant to choose the next step in the solution. A control sentence can not decide that a sequence of steps should be pursued, but can only select a single step. Therefore, this approach ties the program down to an extremely "deliberate" control process.

A second method of control that we have implemented for PROTEAN incorporates experimental results with control sentences and imposes more structure on the problem solving sequence. The results in [Garvey *et al.*, 1987] showed that certain modifiers in the control sentence were more important than others, and usually led to better performance. For instance, in the case of selecting an anchor, the number and distribution of constraints to other objects is the most important variable (as captured by the modifier **CONSTRAINING**). We compiled this rule into a new knowledge source which procedurally defined the criteria for choosing an anchor by examining properties of the initial constraint network. We used similar results from our own studies to compile a procedure for deciding which objects should be introduced into a partial arrangement with the ANCHOR action versus the APPEND action. We added this information to the new knowledge source and were left with a domain problem solving KS which chooses the best global anchor, the best anchorees to introduce into the global partial arrangement as well as other "secondary" partial arrangements with which to define local geometries.

This KS, therefore, produces a partial plan for solution of the problem (shown in Figure 4). As a result of compiling PROTEAN's declarative control sentences, Helix-5 can be uniquely identified without further search to be the anchor. In addition, Helices 1,3,8,9 and 10 are designated as anchorees. Other objects are to be introduced into the space of Helix-5 with an APPEND action. In order to implement this plan, we also added a single control statement that favors KSARs mentioned in the plan. This control statement simply checks to see if the KSAR appears in the plan or not, and replaces other control declarations (like the ORIENT control sentence shown previously) that require interpretation and matching. Our plan allows us to remove control sentences that address control issues 1,2,3 and 4 as listed in section II-B. Three points should be emphasized:

1. Our method for compilation has two steps:

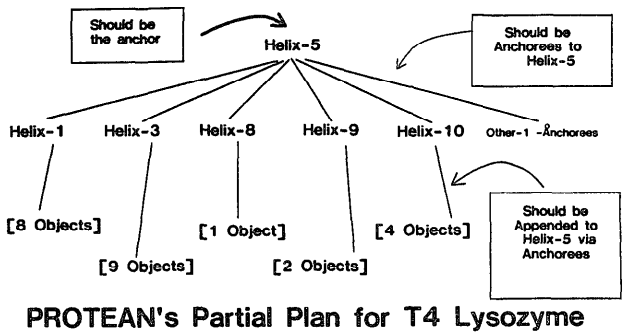

Figure 4: A graphical depiction of the partial plan for solution of T4 Lysozyme.

   (a) **Manual Static Compilation**
      - Reformulate declarative *strategic* knowledge (derived from experimentation) into *domain* procedures for solving a problem. These procedures are contained in new domain knowledge sources and specify the compiled *criteria* for choosing objects and actions during problem solving.
      - Replace all reformulated declarative strategic knowledge with a single strategic statement that chooses domain actions mentioned in the plan whenever they become executable.

   (b) **Dynamic Compilation in Context** Execute the procedure *in the context of a problem statement* in order to actually select objects and actions that constitute a partial plan for solution. The instantiated plan is used to guide control decision making.

2. By compiling strategic knowledge, we have decided to make some control decisions in advance. The "compiled" decisions are based on evaluation of the *static* properties of the objects in the problem and the domain problem solving actions. They should not depend critically on dynamic properties of the problem. If unanticipated problems occur in implementation of the plan, this decision may prove to be extremely expensive. It is therefore important to have confidence in the declarative strategic sentences that are compiled. In our compilation of knowledge we have not altered any of the other knowledge sources for problem solving; we have just added one domain KS for planning, and a control declaration that requires KSARs that implement the plan to be executed before others. Thus, the compilation step is modular, relatively non-destructive to system integrity, and decreases the number of declarative sentences that must be interpreted and matched.

3. Having an overview of the global solution strategy also offers opportunities which are not available without a plan. For example, the plan produced by our procedure immediately suggests subtasks for parallel execution. Each of the secondary partial arrangements of Figure 4 represents an independent constraint network that can be brought to equilibrium in isolation from the others.

When the plan is produced and instructions for following the plan are added, the nature of control changes significantly. The issue of choosing an anchor, for example, is not a

significant control issue any longer: it has been moved into the procedural detail of a problem solving knowledge source. However, the exact order in which to perform YOKE operations still remains unresolved. Thus, the selection of feasible yoking actions has been left in a declarative, opportunistic framework. The plan thus leaves significant details (i.e., the order of yokes) unresolved until run-time interpretation of control knowledge sources, while fixing some details in the compiled steps. When all such strategy knowledge has been compiled into domain knowledge, then the strategy becomes procedurally defined by the sequence of these domain knowledge sources and there is little flexibility for testing alternative strategies.

III. Example: PROTEAN's Problem Solving Behavior with a Plan

In order to illustrate the behavior of PROTEAN using a solution plan, we present the results of the method when applied to the protein phage T4 Lysozyme. PROTEAN processes its input to define 37 superatoms into which the protein can be divided (as suggested by experimental data). In addition, PROTEAN creates a **constraint set** for each pair of objects between which there are distance constraints. Not all objects have constraints with other objects, so there is a total of 119 constraint sets (out of the total possible $(37^2 - 37)/2 = 666$). A graphical depiction of the constraint matrix is shown in Figure 5.

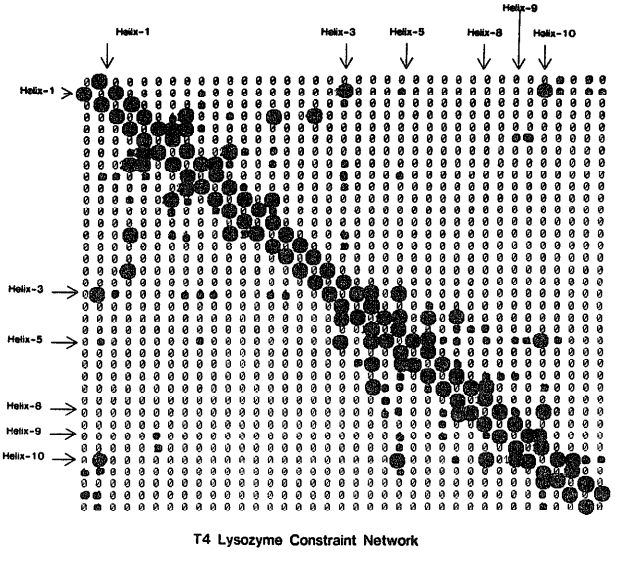

Figure 5: Matrix depiction of the constraint network in T4 Lysozyme.

The objects occur in chemically linked sequence and are numbered 1 to 37, from left to right and top down. The approximate strength of the constraint set, C_{ij}, is indicated by the size of the spot at matrix position ij (or ji). The constraint rows for key subunits are labelled. The matrix shows two large clusters of constraints in this system. There are generally strong constraints between neighboring subunits, but few objects have strong constraints to distant objects. It is clear that the strategy decisions outlined in section II-B are not obvious, and require reasoning and analysis of the network.

A trace of the problem solving behavior of PROTEAN is useful in understanding how strategic reasoning and domain actions combine to produce useful problem solving behavior. For any given cycle, BB1 may follow a compiled decision to follow the plan (D1), may reason out a strategic decision about the best action (D2), or may perform the domain actions specified by D1 or D2 (A).

CYCLE	TYPE OF REASONING		DECISIONS/ACTIONS
0	CONTROL	(D1)	Decides to Run the KS which examines the problem and produces a plan.
1	DOMAIN	(A)	Plan algorithm is run, HELIX5 is chosen as the anchor, 12 objects are designated anchorees, and 14 objects are designated appendees (See Figure 4).
2	CONTROL	(D1)	Decides to implement the plan from cycle 1 by automatically favoring KSARs which directly implement pieces of the plan.
3-30	DOMAIN	(A)	Partial Arrangement 1 (PA1) is established and oriented around HELIX5. Anchorees are introduced into PA1 and ANCHORed to HELIX5.
31	CONTROL	(D2)	Decides to YOKE accessible volumes determined in previous cycles. NOTE: there is no plan specification for this, so it is done opportunistically with declarative control.
32-110	DOMAIN	(A)	YOKES are favored between objects that are LARGE, have STRONG constraint sets, and have BIG-RELATIVE-DIFFERENCE in the size of their location tables. They continue until the constraint network within the partial arrangement reaches equilibrium.
111	CONTROL	(D1)	Decides to establish the secondary anchor spaces, ORIENTed around the secondary anchors as specified by the plan, and ANCHOR the appendees as specified.
112-200	DOMAIN	(A)	Carries out plan for secondary partial arrangements by ANCHORing appendees to secondary anchors.
201	CONTROL	(D2)	Decides to YOKE objects in secondary PAs in order to reduce location table size.
202-240	DOMAIN	(A)	Opportunistically YOKES objects.
241	CONTROL	(D1)	Decides to APPEND appendees into main PA1 as specified by the plan.
241-250	DOMAIN	(A)	APPENDs appendees into main PA1.
251	CONTROL	(D2)	Decides to continue YOKING new location tables in PA1 with previously yoked location tables from cycles 32-110.
252-400	DOMAIN	(A)	YOKES opportunistically until network equilibrium is reached and all location tables are at minimum. At this point, backtrack search CONSOLIDATION can be performed.

About half of the control decisions are compiled in this example, and about half the resulting domain actions follow directly from them, rather than by interpreting and matching high level predicates. The plan shown in Figure 4 is partial because there are significant numbers of reasoning cycles in which it makes no prescription for action (cycles 32-110, 202-240, and 252-400), and "best first" strategies must be used. However, the structure imposed on problem solving by the initial plan is strong enough to provide a clear procedural outline. We can continue to use a purely declarative control structure for testing and improving the plan if weaknesses are discovered.

IV. Related Work

Full descriptions of BB1, PROTEAN and our initial control strategies can be found in [Altman and Jardetzky, 1986, Brinkley *et al.*, 1986, Hayes-Roth, 1985, Hayes-Roth *et al.*, 1986b]. The theme of transformation from declarative to procedural specification arises in many artificial intelligence programming

efforts. Our work stresses the usefulness of the partial plan as an intermediate step.

The EMYCIN system contains a rule compiler that maps domain rules into a decision tree [van Melle, 1980]. The decision tree is a fully elaborated plan for solution of the problem, and as such corresponds to the final stage or our three-phase paradigm. EMYCIN has a static view of how to control evidence gathering (goal-driven, backward chaining). We argue that an intermediate step of compiling control knowledge into domain rules before production of such a decision tree provides a greater flexibility in the development of procedures, since the control strategies used need not be static. HERACLES is an example of another system which uses declarative representations of strategies, and thus could benefit from an intermediate stage of control compilation [Clancey, 1985]. Similarly, meta-knowledge used by systems such as PLANNER's rule filters [Hewitt, 1972] or NASL's choice rules [McDermott, 1978] can be compiled into domain rules to gain efficiency at the expense of flexibility.

Skeletal planning was characterized by Friedland in the MOLGEN work [Friedland, 1979]. Our work uses many of the ideas of heuristic application of a global problem solving strategy. Our plans are partial with respect to the complete sequence of problem solving, but are not generalized to higher level concepts (i.e., they are expressed in the the low level vocabulary of domain actions). In that respect they are similar to Schank's *scripts*, but are *partial* scripts [Schank and Abelson, 1975].

V. Conclusions

The PROTEAN system for geometric constraint satisfaction in the domain of protein structure provides an excellent forum in which to experiment with different strategies. Others have formulated declarative strategies, and we have described here a compilation of parts of these strategies into domain problem solving actions. The compilation of strategic knowledge has lead to a partial plan which focuses problem solving and requires less control deliberation. The plan has been used to determine the structure of T4 Lysozyme, and provides a framework for expansion of the procedural element of strategic reasoning in the future.

Our method works well in domains in which nearly independent strategic decisions can be identified prospectively. Context dependent decisions can be made opportunistically at run time since we perform only partial compilation. This mixture of opportunistic and planned problem solving is especially powerful in domains such as PROTEAN's in which a plan is useful for solving subproblems but opportunism is required to recombine or conjoin the solutions to the subproblems.

References

[Altman and Jardetzky, 1986] R. Altman and O. Jardetzky. New strategies for the determination of macromolecular structure in solution. *J. Biochem*, 100(6):1403–1423, December 1986.

[Brinkley *et al.*, 1986] J. Brinkley, C. Cornelius, R. Altman, B. Hayes-Roth, O. Lichtarge, B. Duncan, B. Buchanan, and Jardetzky O. *Application of Constraint Satisfaction Techniques to the Determination of Protein Tertiary Structure.* Technical Report KSL 86-28, Knowledge Systems Laboratory, Stanford University, March 1986.

[Clancey, 1985] W.J. Clancey. Heracles: representing procedures as abstract metarules. 1985. To appear in 'Computer Expert Systems', M. J. Coombs and L. Bolc, eds. Springer-Verlag, in preparation.

[Davis, 1980] R. Davis. Meta-rules: reasoning about control. *Artificial Intelligence*, 15:179–222, 1980.

[Friedland, 1979] P. Friedland. *Knowledge-based Hierarchical Planning in Molecular Genetics.* PhD thesis, Computer Science Department, Stanford University, September 1979. Report CS-79-760.

[Garvey *et al.*, 1987] A. Garvey, C. Cornelius, and B. Hayes-Roth. *Computational Costs versus Benefits of Control Reasoning.* Technical Report KSL 87-11, Knowledge Systems Laboratory, Stanford University, February 1987. To appear in 'Proceedings of AAAI, 1987'.

[Hayes-Roth, 1985] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[Hayes-Roth *et al.*, 1986a] B. Hayes-Roth, A. Garvey, M.V. Johnson, and M. Hewett. *A Layered Environment for Reasoning about Action.* Technical Report KSL 86-38, Stanford University, November 1986.

[Hayes-Roth *et al.*, 1986b] B. Hayes-Roth, B.G. Buchanan, O. Lichtarge, M. Hewett, R. Altman, J. Brinkley, C. Cornelius, B. Duncan, and O. Jardetzky. Protean: deriving protein structures from constraints. In *Proceedings of the AAAI*, pages 904–909, Morgan Kaufmann Publishers, Inc., 1986.

[Hewitt, 1972] C. Hewitt. *Description and theoretical analysis using schemata of PLANNER, a language for proving theorems and manipulating models in a robot.* Technical Report TR-258, AI Laboratory, M.I.T., 1972.

[Mackworth, 1977] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[McDermott, 1978] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[Schank and Abelson, 1975] R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1975.

[Schulman and Hayes-Roth, 1987] R. Schulman and B. Hayes-Roth. *ExAct: A Module for Explaining Actions.* Technical Report KSL-87-8, Knowledge Systems Laboratory, Stanford University, February 1987.

[van Melle, 1980] W. van Melle. *A domain-independent system that aids in constructing knowledge-based consultation programs.* PhD thesis, Computer Science Department, Stanford University, June 1980.