Computational Costs versus Benefits of Control Reasoning[1]

Alan Garvey, Craig Cornelius, and **Barbara Hayes-Roth**
Knowledge Systems Laboratory
Stanford University

Abstract

We assess the computational costs and benefits of control reasoning in the PROTEAN system, which is built in the BB* environment. We experimentally manipulate PROTEAN's control knowledge and analyze differences in total problem-solving time as a function of several component times. Our results demonstrate and explain the utility of control reasoning. They also illustrate the importance of experimental investigation and the utility of the BB* environment for conducting such investigations.

I. Reasoning About Control

Control reasoning–deciding which of many possible actions to perform at each point in time–can affect the probability that a system achieves its goal, the resources it consumes, and the side effects it produces. Focusing on potential advantages, some researchers have developed general problem-solving architectures that support control reasoning [Durfee and Lesser, 1986, Erman *et al.*, 1981, Genesereth and Smith, 1982, Hayes-Roth, 1985]. On the other hand, even the most beneficial control reasoning entails computational costs[Durfee and Lesser, 1986, Hayes-Roth and Lesser, 1977, Smith and Genesereth, 1985]. The question is: What is the computational cost/benefit tradeoff for control reasoning?

We address this question empirically with the PROTEAN system[Hayes-Roth *et al.*, 1986b], which is implemented in the BB* environment[Hayes-Roth *et al.*, 1986a].

II. Overview of BB* and PROTEAN

The PROTEAN system is implemented within the BB* environment: a hierarchy of knowledge modules, which can be configured and layered upon one another to build systems. As described below, PROTEAN's knowledge about proteins is layered upon ACCORD's knowledge about assembling arrangements of objects to satisfy constraints, which is layered upon the BB1 blackboard control architecture[Hayes-Roth, 1985].

A. The BB1 Blackboard Control Architecture

The BB1 blackboard control architecture provides a uniform mechanism for reasoning about problems and problem-solving actions. Functionally independent *knowledge sources (KSs)* cooperate to solve problems by recording and modifying solution elements in a global data structure, the *blackboard. Domain KSs* solve domain problems on the *domain blackboard. Control KSs* construct control plans for the system's own actions on the *control blackboard.* All KSs, when triggered, generate *knowledge source activation records (KSARs)* that compete for scheduling priority.

The BB1 execution cycle has three steps: (a) The *interpreter* executes the action of the scheduled KSAR, thereby changing the blackboard. (b) The *agenda-manager* adds KSARs to the agenda for all KSs triggered by the blackboard changes and rates each one against the current control plan. (c) The *scheduler* chooses the highest-rated KSAR to execute its action next. If it schedules a control KSAR, that KSAR may change the criteria used to rate pending KSARs on subsequent cycles.

Given this architecture, an application system can exploit the full power of the blackboard architecture to construct and follow control plans for its own actions in real time. For example, it can incrementally refine a general strategy as a sequence of specific objectives. It can pursue multiple plans simultaneously. It can integrate opportunistic, goal-driven, and data-driven objectives in its plans[Johnson and Hayes-Roth, 1986]. It can modify, interrupt, depart from, resume, or abandon plans.

B. The ACCORD Framework

ACCORD provides a domain-independent framework for performing *arrangement-assembly tasks.* Within ACCORD, a problem-solver defines several *partial arrangements*, each comprising some of the objects and constraints specified in a problem. It declares one object the *anchor* and positions other objects (*anchorees*) relative to it. It reduces the *family* of legal positions for each anchoree by *anchoring* it with constraints to the anchor and *yoking* it with constraints to other anchorees. Eventually, the problem solver *integrates* multiple partial arrangements with constraints among their constituent objects.

To support arrangement assembly, ACCORD provides: (a) a skeletal concept network in which to define domain-specific objects and constraints; (b) a vocabulary of partial arrangements (e.g., *anchor, anchoree*); (c) a type hierarchy of

[1]This research was supported by the following grants: NIH Grant RR-00785; NIH Grant RR-00711; Boeing Grant W266875; NASA/Ames Grant NCC 2-274; DARPA Contract N00039-83-C-0136; ONR Contract N00014-86-K-0652. We thank Micheal Hewett, M. Vaughan Johnson Jr., Robert Schulman and Jeff Harvey for their work on BB1. We thank Russ Altman, Jim Brinkley, Bruce Duncan, Olivier Lichtarge, John Brugge, and Oleg Jardetzky for their work on PROTEAN. Special thanks to Bruce Buchanan and Ed Feigenbaum for sponsoring the work within the Knowledge Systems Laboratory.

assembly actions, events, and states (e.g., *do-anchor is-a do-position action*); (c) linguistic templates for instantiating actions, events, and states (e.g., *Do-anchor anchoree to anchor in partial-arrangement with constraints*).

ACCORD enables an application system to reason about its problem-solving actions and control plans in an interpretable, English-like representation. For example, PROTEAN represents one of its actions as:

Do-Anchor Helix-1-1 to Helix-2-1 in PA1 with NOE6.

It represents one of its control decisions as:

Do-Position Long Helix in PA1 with Strong Constraint.

BB1 determines that the action matches the control decision because:

Do-Anchor is-a Do-Position action.
Helix-1-1 is Long.
Helix-1-1 is-a Helix.
PA1 is PA1.
NOE6 is Strong.
NOE6 is-a Constraint.

Finally, BB1 translates the action into its executable language of blackboard modifications.

C. The PROTEAN System

1. Knowledge

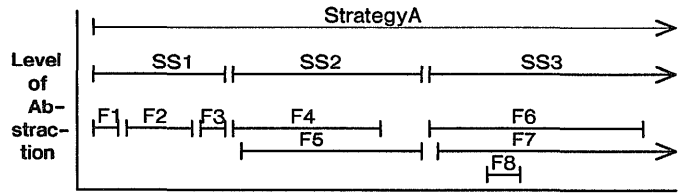
PROTEAN attempts to identify the three-dimensional conformations of proteins based on a variety of constraints, using four kinds of knowledge. It instantiates ACCORD's concept network with biochemistry objects (e.g., *Helix is-a Object*) and constraints (e.g., *NOE is-a Constraint*). It specifies domain KSs that generate feasible problem-solving actions. For example, one KS specifies:

Trigger: Did-Position Anchoree (The-Object)
in PA (The-PA)
Context: For The-Partner in:
Includes The-PA Anchoree (The-Partner)
Constrains The-Object The-Partner
with Constraints (The-Constraints)
Action: Do-Yoke The-Object with The-Partner
in The-PA with The-Constraints

It specifies a *geometry system (GS)* [Brinkley *et al.*, 1986] (discussed below) that performs the numerical operations underlying certain actions. It specifies control KSs that generate the control plan in Figure 1 (discussed below).

2. Geometry System

PROTEAN's GS performs two operations. To support anchoring actions, the GS *searches space*, generating all possible locations for an anchoree (at some resolution) that satisfy the anchoring constraints. Since six parameters specify the position and orientation of an object in space, both the computation time to search space and the number of locations returned increase roughly as the sixth power of the sampling resolution. PROTEAN determines resolution by instructing the GS to: (a) begin searching at low resolution; and (b) repeat the search at progressively higher resolutions until it returns a *threshold* number of locations. To support yoking and other positioning actions, the GS *prunes locations*, testing each location against the specified constraints and returning all locations that satisfy them. Since yoking compares each pair of locations for two previously anchored objects, yoking time increases as the twelfth power of the resolution.



StrategyA: Assemble One Partial-Arrangement

Sub-Strategy1 (SS1): Define one Partial-Arrangement

Sub-Strategy2 (SS2): Position Structured-Secondary-Structure

Sub-Strategy3 (SS3): Position Secondary-Structure

Focus1 (F1): Create Anyname

Focus2 (F2): Include Secondary-Structure in PA1

Focus3 (F3): Orient PA1 about long contraining constrained Structured-Secondary-Structure

Focus4 (F4): Anchor *{o1}* Structured-Secondary-Structure to Helix1-1 in PA1 with *{c}* Constraint-Set

Focus5 (F5): Yoke several *{o3}* Structured-Secondary-Structure in PA1 with *{c}* Constraint-Set

Focus6 (F6): Anchor *{o2}* Random-Coil to Helix1-1 in PA1 with *{c}* Constraint-Set

Focus7 (F7): Yoke several *{o3}* Secondary-Structure in PA1 with *{c}* Constraint-Set

Focus8 (F8): Restrict Secondary-Structure in PA1 with Constraint-Set

Figure 1: Basic PROTEAN Control Plan

3. Control Plan

PROTEAN's *strategy* (see Figure 1) comprises a sequence of three *sub-strategies*, each comprising a sequence or set of *foci*. (The next section explains the bracketed characters in Figure 1.) During SS1, PROTEAN creates a partial-arrangement, includes objects in it, and orients it around a particular anchor. During SS2, PROTEAN positions structured objects (*alpha-helices* and *beta-strands*) by anchoring and yoking them. During SS3, PROTEAN positions all objects (including non-structured *coils*) by anchoring and yoking them. Whenever PROTEAN generates an intractably large number of locations for an anchoree, it introduces an *opportunistic* focus (e.g., F8) to *restrict* (statistically sample) the locations.

PROTEAN generated the control plan in Figure 1 for the lac-repressor headpiece protein at low resolution. For other proteins or resolutions, sub-strategies and foci appear and terminate on different cycles.

III. Experimental Manipulations

A. Control Knowledge

We studied four variations on PROTEAN's basic control plan. Strategy A generated the plan in Figure 1. Strategy B introduced the constraint modifier *strong* at points indicated by *c*. Strategy C introduced object modifiers as follows: *long, inflexible, constrained,* and *constraining* at *o1*, *constraining* at *o2*, and *long, inflexible, constraining,* and *recently-reduced* at *o3*. Strategy D introduced all modifiers. For example, here are the four versions of Focus 7:

Strategy A: Yoke several Secondary-Structure in PA1 with Constraint-Set.

Strategy B: Yoke several Secondary-Structure in PA1 with Strong Constraint-Set.

Strategy C: Yoke several Long Inflexible Constraining Recently-Reduced Secondary-Structure in PA1 with Constraint-Set.

Strategy D: Yoke several Long Inflexible Constraining Recently-Reduced Secondary-Structure in PA1 with Strong Constraint-Set.

Modifiers increase the precision with which the strategy discriminates among competing actions. For example, sentence A gives equal ratings to all actions that yoke secondary-structures in partial-arrangement PA1 with **any** constraints, while sentence B gives higher ratings to actions that use **strong** constraints.

Domain experts recommend these particular modifiers to favor positioning actions that rapidly reduce the number of locations for each object. While these modifiers don't affect PROTEAN's ultimate solution, they should reduce the number of positioning actions it performs and the cost of later actions. On the other hand, they should increase the cost of each rating action. The question is: Does the benefit of performing fewer, more effective positioning actions outweigh the cost of identifying those actions?

B. Proteins

We compared PROTEAN's four strategies on each of two proteins: the lac-repressor headpiece and myoglobin. They differ on: size (51 amino acids vs. 153), number of structured objects (3 versus 5), number of NOE constraints (17 vs. 21), sequence of amino acids, and pattern of constraints.

C. Resolution

We compared PROTEAN's four strategies on the lac-repressor headpiece at each of three thresholds (see II.C.2). The *high* resolution threshold was double the *medium* resolution threshold, which was double the *low* resolution threshold. Compared to low resolution medium and high resolution produced four times as many locations for Helix2 and Helix3. High resolution also produced four times as many locations for Coil4.

IV. Analysis

We analyzed our results with this model:

```
Total Time = Total Symbolic Reasoning Time
           + Total Rating Time + Total GS Time.

Total Symbolic Reasoning Time =
      F(Total Number of Cycles
         & Number of KSARs per Cycle
         & Identities of KSARs per Cycle)

Total Rating Time =
      F(Total Number of Cycles
         & Number of KSARs per Cycle
         & Rating Time per KSAR)

Total GS Time = F(GS Operations Performed
         & Threshold)
```

We assess the effects of each knowledge strategy by comparing it to strategy A as follows:

1. Does the control knowledge affect the identities or number of actions PROTEAN schedules?
2. Do differences in scheduling decisions affect the efficiency (total GS time or total symbolic reasoning time) of PROTEAN's problem solving?
3. Does the control knowledge affect the cost (rating time per KSAR) of PROTEAN's scheduling decisions?
4. Do the combined effects produce a net computational efficiency (total time) in PROTEAN's performance?

V. Results

A. Results for the Lac-Repressor Headpiece

Table 1 shows PROTEAN's performance on the lac-repressor headpiece. Notice that the cost of control knowledge is negligible (.05-.64 second increase in rating time per KSAR) compared to total time (1587-5171 seconds). It does not significantly affect PROTEAN's overall efficiency.

The top panel of Table 1 shows the complete results. As indicated by total time, Strategy B produced a net efficiency compared to Strategy A. Strategy B reduced PROTEAN's total number of actions by nine, thereby reducing both GS and symbolic reasoning times. These effects outweighed the small increase in rating time. By contrast, Strategy C produced a net inefficiency. It reduced the number of actions by one, slightly reducing GS time. But increased symbolic reasoning and rating times outweighed this savings. Strategy D, which included all modifiers, combined Strategy B's reductions in GS and symbolic reasoning times with Strategy C's increase in rating times, producing an intermediate net efficiency.

The middle panel of Table 1 shows the results for substrategy SS2. Here PROTEAN anchored and yoked two helices relative to a third, performing exactly the same actions (in slightly different orders) under all four strategies. As a consequence, the knowledge strategies did not affect GS or symbolic reasoning times, but did slightly increase rating times. The strategies did not significantly affect total times.

The bottom panel of Table 1 shows the results for substrategy SS3. Here PROTEAN anchored the four coils and yoked all of the anchorees to one another. These results parallel and actually determine the complete results in the top panel of Table 1. Strategy B reduced PROTEAN's total number of actions by nine, thereby reducing GS and symbolic reasoning times. While slightly increasing ratings times, Strategy B produced a net efficiency. Strategy C reduced the number

	A. No Modifiers	B. Constraint Modifiers	C. Object Modifiers	D. All Modifiers
Costs During All Sub-strategies				
Total Time	4881	4265	5171	4281
Number of Cycles	87	78	86	78
GS Time	3322	2858	3294	2864
Symbolic Reasoning Time[a]	1381	1240	1561	1191
Average KSAR Rating Time	0.35	0.41	0.87	0.93
Costs During Sub-strategy 2				
Total Time	2266	2271	2270	2268
Number of Cycles	7	7	7	7
GS Time	2186	2186	2186	2186
Symbolic Reasoning Time[a]	71	73	70	68
Average KSAR Rating Time	0.35	0.40	0.85	0.91
Costs During Sub-strategy 3				
Total Time	2215	1587	2518	1635
Number of Cycles	37	28	36	28
GS Time	1136	666	1108	678
Symbolic Reasoning Time[a]	928	768	1123	761
Average KSAR Rating Time	0.34	0.41	0.89	0.95

All times are in seconds.

[a]This is all symbolic computing time, except rating time.

Table 1: Computational Costs of Four Strategies for Assembling the Lac-Repressor Headpiece

	A. No Modifiers	B. Constraint Modifiers	C. Object Modifiers	D. All Modifiers
Costs During All Sub-strategies				
Total Time	13930	11985	12460	11816
Number of Cycles	116	104	111	103
GS Time	7278	6898	6275	6304
Symbolic Reasoning Time[a]	6018	4506	5500	4795
Average KSAR Rating Time	0.35	0.42	1.45	1.51
Costs During Sub-strategy 2				
Total Time	5062	5327	5410	5100
Number of Cycles	22	21	23	20
GS Time	4383	4699	4595	4453
Symbolic Reasoning Time[a]	602	559	687	524
Average KSAR Rating Time	0.35	0.41	1.70	1.74
Costs During Sub-strategy 3				
Total Time	7536	5413	5869	5467
Number of Cycles	43	32	37	32
GS Time	2895	2199	1680	1851
Symbolic Reasoning Time[a]	4121	2735	3665	3056
Average KSAR Rating Time	0.35	0.42	1.20	1.28

All times are in seconds.

[a]This is all symbolic computing time, except rating time.

Table 2: Computational Costs of Four Strategies for Assembling Myoglobin

of actions by one, slightly decreasing GS time, but increasing symbolic reasoning time. Increasing rating times as well, Strategy C produced a net inefficiency. Strategy D combined these effects to produce an intermediate net efficiency.

B. Results for Myoglobin

Table 2 shows PROTEAN's performance on myoglobin. Again, the cost of control knowledge is negligible (.07-1.39 seconds increase in rating time per KSAR) compared to total time (5062-13930 seconds). It does not significantly affect PROTEAN's overall efficiency.

The top panel of Table 2 shows the complete results. As indicated by total time, all three knowledge strategies reduced the number of actions PROTEAN performed, reducing both GS and symbolic reasoning times. These effects outweighed increases in rating times, producing a net advantage in efficiency. As for the lac-repressor headpiece, Strategy B's constraint modifiers were more effective than Strategy C's object modifiers. Here, however, Strategy D's combined modifiers produced the greatest efficiency.

The middle panel of Table 2 shows the results for substrategy SS2. Here PROTEAN positioned five structured objects. All three knowledge strategies produced about the same number of actions, but increased GS time. PROTEAN's scheduling records show that PROTEAN performed several specific yoking actions earlier under the knowledge strategies than it did under Strategy A. At the earlier times, these particular actions required more expensive GS operations than they required later in problem solving, but did not reduce the total number of actions required to solve the problem. These costs, combined with increased ratings times, produced net inefficiencies in total time for all three knowledge strategies. We are conducting experiments that combine control knowledge of the cost of positioning actions with current control knowledge of their effectiveness.

The bottom panel of Table 2 shows the results for substrategy SS3. Here PROTEAN positioned the five structured objects and four coils. All three knowledge strategies reduced the number of actions PROTEAN performed, substantially reducing GS and symbolic reasoning times. These savings outweighed the increased cost of rating, producing a net efficiency in total time.

C. Effects of Resolution

Table 3 shows PROTEAN's performance on the lac-repressor headpiece during sub-strategy SS3 under all four strategies at each of three resolutions. Because higher resolutions entail more expensive GS computations, the knowledge strategies produce larger net efficiencies. Thus, maximum savings in total time range from 628 seconds at low resolution to 1771 seconds at medium resolution to 4110 seconds at high resolution.

Table 3 also shows an interaction between strategy and resolution. Strategies B and D produced essentially the same effects at all resolutions. They reduced the number of actions PROTEAN performed, thereby reducing GS and symbolic reasoning times. In all cases, these savings outweighed the increased rating times, producing net efficiencies in total time. By contrast, Strategy C produced different effects at different resolutions. At low resolution, Strategy C reduced the number of actions by only one, slightly reduced GS time, increased symbolic reasoning time, and increased rating time. It produced a net inefficiency in total time. At medium and high

	A. No Modifiers	B. Constraint Modifiers	C. Object Modifiers	D. All Modifiers
Costs During Sub-strategy 3 at Low Resolution				
Total Time	2215	1587	2518	1635
Number of Cycles	37	28	36	28
GS Time	1136	666	1108	678
Symbolic Reasoning Time[a]	928	768	1123	761
Average KSAR Rating Time	0.34	0.41	0.89	0.95
Costs During Sub-strategy 3 at Medium Resolution				
Total Time	5268	3831	4059	3497
Number of Cycles	37	31	32	28
GS Time	4191	2718	3140	2550
Symbolic Reasoning Time[a]	917	895	766	759
Average KSAR Rating Time	0.34	0.41	0.89	0.95
Costs During Sub-strategy 3 at High Resolution				
Total Time	9956	7729	5846	5925
Number of Cycles	38	33	29	28
GS Time	8794	6746	4819	4939
Symbolic Reasoning Time[a]	997	840	847	790
Average KSAR Rating Time	0.34	0.41	0.89	0.95

All times are in seconds.

[a]This is all symbolic computing time, except rating time.

Table 3: Computational Costs of Sub-strategy 3 at Three Resolutions for Four Strategies for the Lac-Repressor Headpiece

resolution, Strategy C reduced the number of actions by five and nine, thereby substantially reducing GS and symbolic reasoning times. These savings outweighed increased rating time, producing a net efficiency in total time. Although we do not fully understand the GS properties that lead to this interaction, we have a hypothesis. Strategy C includes the only modifier, *recently-reduced*, that is sensitive to intermediate solution states. Perhaps the GS produces results that differentially satisfy this modifier at different resolutions. We are investigating this hypothesis.

D. Effects of Control on Symbolic Reasoning *Per Se*

PROTEAN differs from many knowledge-based systems in its dependence upon expensive computations performed by the geometry system. As discussed above, the three knowledge strategies produce substantial efficiencies in PROTEAN's performance largely by producing savings on GS computations. However, the knowledge strategies produce net efficiencies in performance independent of the cost of GS computations. Table 4 shows net symbolic reasoning time (total time - GS time) for all cases in which the knowledge strategies produced a net overall efficiency. In all cases but one, the knowledge strategies produced more efficient symbolic reasoning *per se*, simply because they allowed PROTEAN to solve problems in fewer problem-solving cycles.

	A. No Modifiers	B. Constraint Modifiers	C. Object Modifiers	D. All Modifiers
Lac-Repressor at Low Resolution				
Net Symbolic Reasoning Time	1559	1407	—	1417
Lac-Repressor at Medium Resolution				
Net Symbolic Reasoning Time	1539	*1540*	1335	1402
Lac-Repressor at High Resolution				
Net Symbolic Reasoning Time	1622	1416	1470	1447
Myoglobin at Low Resolution				
Net Symbolic Reasoning Time	6652	5087	6185	5512

All times are in seconds.

Table 4: Net Symbolic Reasoning Time for Cases where Knowledge Strategies Produced Net Efficiency

E. Effects of Control on Identified Protein Structures

The four strategies examined in these experiments had no effect on the protein structures PROTEAN identified. At a given level of resolution, it identified exactly the same structure for the lac-repressor under all four strategies. Similarly, it identified exactly the same structure for myoglobin under all four strategies.

VI. Implications

Our results confirm that intelligent control reasoning can induce computational efficiency in AI systems. In particular, we found that:

- Control knowledge, including object and constraint modifiers, reduces total problem-solving time by reducing the number of actions performed, GS time, and symbolic reasoning time.
- The cost of using control knowledge–rating actions against modifiers–is negligible compared to total problem-solving time.
- Control knowledge is most effective when the scheduler chooses among many possible actions and its choices alter the number or cost of subsequent actions.
- Constraint modifiers usually reduce GS time more effectively than object modifiers.
- Sometimes the most effective actions entail disproportionately expensive GS operations.
- Control knowledge produces larger savings at higher GS resolutions.
- Modifiers that measure intermediate solution states may operate more effectively at higher GS resolutions.

We plan to incorporate these results into PROTEAN's knowledge base, so that it can decide which modifiers to use in particular problem-solving situations.

We conjecture that PROTEAN's control knowledge would have similar effects in other "arrangement-assembly systems." For example, the SIGHTPLAN system[Tommelein *et al.*, 1987] designs construction site layouts by assembling arrangements of

construction areas and equipment in a two-dimensional spatial context. Since SIGHTPLAN also is implemented in BB*, we can easily determine whether object and constraint modifiers analogous to those defined for PROTEAN have similar effects on its efficiency.

Speculating more broadly, we conjecture that control knowledge of the sort used in these experiments (modifiers inserted into focus decisions) would improve the efficiency of any application in which: (a) actions require expensive computations; and (b) choice of actions affects the number or cost of subsequent actions.

The experiments illustrate a method for analyzing the utility of control knowledge. Here we introduced modifiers to the strategic parameters of a basic control plan. In new experiments, we manipulate the structure of the control plan itself. The BB* environment facilitates these experiments. BB1 provides tools for building control plans of any hierarchical/heterarchical complexity. ACCORD provides a language for representing and reasoning about plans. Both BB1 and ACCORD permit modular variations on the form and content of control plans.

In our investigation of control reasoning, we also need to assess costs and benefits at the architectural level. Could another architecture exploit the necessary control knowledge more efficiently than BB1? How do alternative architectures compare on: ease of system development, clarity of knowledge representation, support for explanation capabilities, and support for learning capabilities. We are conducting experiments that address these questions.

Finally, our experience in conducting these experiments argues strongly for experimental investigation of theoretical assertions. Although we thoroughly understand the operation of BB1, ACCORD, PROTEAN, and GS, we could not reliably predict important details of their performance. For example: (a) we mistakenly expected increases in rating time to substantially limit the net advantages of control knowledge; and (b) we still do not fully understand why Strategy C produced net inefficiency at low resolution, but net efficiency at medium and high resolution. Given the inherent complexity of contemporary AI systems and the weakness and potential bias of human efforts to anticipate their behavior, experimental methods must play a key role in our research.

References

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

[Durfee and Lesser, 1986] E.H. Durfee and V.R. Lesser. Incremental planning to control a blackboard-based problem solver. *Proceedings of the Fifth National Conference on Artificial Intelligence*, :58–64, 1986.

[Erman *et al.*, 1981] L.D. Erman, P.E. London, and S.F. Fickas. The design and an example use of Hearsay-III. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, :409–415, 1981.

[Genesereth and Smith, 1982] M.R. Genesereth and D.E. Smith. *Meta-level architecture.* Technical Report HPP-81-6, Stanford University, 1982.

[Hayes-Roth, 1985] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence Journal*, 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, 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 structure from constraints. *Proceedings of the AAAI*, 1986.

[Hayes-Roth and Lesser, 1977] F. Hayes-Roth and V.R. Lesser. Focus of attention in the Hearsay-II speech understanding system. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, :27–35, 1977.

[Johnson and Hayes-Roth, 1986] M.V. Johnson and B. Hayes-Roth. *Integrating Diverse Reasoning Methods in the BB1 Blackboard Control Architecture.* Technical Report KSL-86-76, Stanford University, 1986.

[Smith and Genesereth, 1985] D.E. Smith and M.R. Genesereth. Ordering conjunctive queries. *Artificial Intelligence*, 25:171–215, 1985.

[Tommelein *et al.*, 1987] I.D. Tommelein, R.E. Levitt, and B. Hayes-Roth. Using expert systems for the layout of temporary facilities on construction sites. *CIB W-65 Symposium, Organization and Management of Construction, Berkshire, U.K*, 1987.