December 1984

Report No. STAN-G-84-1034
Also numbered: HPP-84-16

BB1: An architecture for blackboard systems that control, explain, and learn about their own behavior

by

Barbara Hayes-Roth

Department of Computer Science

Stanford University
Stanford, CA 94305

Heuristic Programming Project
Report No. 84-16

December 1984

BB1: An architecture for blackboard systems that control, explain, and learn about their own behavior'

Barbara Hayes-Roth

December, 1984

[1]This work was supported by a DARPA grant to the Heuristic Programming Project. Mtke Hewett. Jeff Harvey, Nelleke Aiello, and Glen Pearson implemented parts of BB1.

Abstract

BB1 implements a domain-independent “blackboard control architecture” for AI systems that control, explain, and learn about their own problem-solving behavior. A BB1 system comprises: a user-defined domain blackboard. a pre-defined control blackboard, user-defined domain and control knowledge sources, a few generic control knowledge sources, and a pre-defined basic control loop. The architecture’s run-time user interface provides capabilities for: displaying the blackboard. knowledge sources. and pending knowledge source actions. recommending an action for execution, explaining a recommendation accepting a user’s override, executing a designated action, and running without user intervention. BB1 supports a variety of control behavior ranging from execution of pre-defined control procedures to dynamic construction and modification of complex control plans during problem solving. It explains problem-solving actions by showing their roles in the underlying control plan. It learns new control heuristics from experience, applies them within the current problem-solving session, and uses them to construct new control plans in subsequent sessions.

1. Introduction

BB1 is a domain-independent architecture for building AI systems that control their own problem-solving behavior, explain their problem-solving actions in terms of an underlying control plan. and learn new control heuristics from experience. BB1 achieves these objectives with a "blackboard control architecture" [6]. Like the standard blackboard architecture [2], the blackboard control architecture provides independent knowledge sources that record solution elements in a structured database--the blackboard-under the control of a scheduler. It differs from the standard architecture in its explicit treatment of control problem-solving. The blackboard control architecture provides explicit domain and control blackboards and knowledge sources. it provides a simple scheduler that decides which domain and control knowledge sources to execute by adapting to control heuristics currently recorded on the control blackboard.

The paper is organized as follows. Section 2 discusses the control. explanation. and learning objectives and the blackboard control architecture's approach to achieving them. Section 3 shows how to build an application system in BB1 using examples from PROTEAN [1], an application system that elucidates protein structure. Section 4 presents a trace from PROTEAN, to illustrate BB1's control, explanation, and learning behaviors. Section 5 compares BB1 to two other blackboard architectures, HEARSAY-III and AGE. Section 6 discusses the costs and benefits of using BB1.

2. Behavioral Objectives and the Blackboard Control A rchitectu re

2.1 Dynamic Control Planning

In attempting to solve a domain problem, an AI system performs a series of problem-solving actions. Each action is triggered by data or previously generated solution elements, applies some knowledge source from the problem domain and generates or modifies a solution element. At each point in the problem-solving process, several such actions may be possible. The control problem is: which of its potential actions should an AI system perform at each point in the problem-solving process?

BB1's blackboard control architecture treats the control problem as a dynamic planning problem. Control knowledge sources, operating concurrently with domain knowledge sources, construct, modify, and execute explicit control plans out of modular control heuristics on a structured control blackboard. A simple scheduler, which selects both domain and control knowledge sources for execution, has no control knowledge of its own. Instead, it adapts its scheduling behavior to the control plan currently recorded on the control blackboard. Within this architecture, BB1 systems perform the following control behaviors.

1. They make explicit control decisions that determine which problem-solving actions they perform at each point in the problem-solving process.

2. They decide what actions to perform by reconciling independent decisions about actions they should perform and actions they can perform.

3. They adopt variable grain-size control heuristics, including global strategies {e.g., First anchor all pieces of secondary structure in partial solutions: Then refine the most credible partial solutions), local objectives (e.g.. Fill in gap g in the current solution), and general scheduling policies (e.g., Exploit the most reliable knowledge sources).

4. They adopt control heuristics that focus on whatever action attributes are useful in the current problem-solving situation. including attributes of their knowledge sources, triggering information, and solution contexts.

5. They adopt. retain, and discard individual control heuristics in response to dynamic problem-solving situations.

6. They decide how to integrate multiple control heuristics of varying importance.

7. They dynamically plan, interrupt, resume, and terminate strategic sequences of actions.

8. They reason about the relative priorities of domain and control actions just as they reason about other control issues.

In sum, BB1 systems typically forgo efforts to predetermine “complete” or “correct” control procedures that anticipate all important problem-solving situations. Instead, they develop control plans incrementally while solving particular domain problems, adapting their behavior to a wide range of unanticipated problem-solving situations. (See [6] for more discussion.)

2.2 Explanation of Problem-Solving Behavior

During efforts to solve a domain problem. an AI system should explain its problem-solving behavior. It should justify actions in terms of the situations that trigger them, the knowledge they use, and the solution elements they generate. It should also show how actions fit into an overall line of reasoning, what specific control heuristics they satisfy, and what alternative actions it considered.

BB1's blackboard control architecture explains a problem-solving action in terms of the dynamic control plan:

1. The current scheduling rule for choosing among feasible actions (e.g., Schedule the highest priority action);

2. The current integration rule for integrating an action’s ratings against multiple control heuristics (e.g., Compute each action’s sum of weighted ratings against operative heuristics);

3. The operative control heuristics (e.g., First anchor all pieces of secondary structure in partial solutions; Then refine the most credible partial solutions. Exploit the most reliable knowledge sources.);

4. The action’s ratings (0- 100) against operative heuristics.

5. The actions priority, computed by applying the integration rule to its ratings.

6. Other actions with the same priority.

2.3 Learning Control Heuristics

During problem-solving, an AI system uses domain knowledge to solve the domain problem and control knowledge to solve the control problem. Although acquiring either kind of knowledge is a formidable task, eliciting control knowledge is especially challenging [4, 5]. Domain experts have difficulty retrieving control knowledge and difficulty distinguishing it from domain knowledge. They frequently produce general control heuristics during questioning, but use more specific heuristics during problem-solving Stimulating experts' retrieval of a comprehensive set of heuristics may require analysis of many example problems that produce no other new knowledge, As a consequence. acquiring comprehensive control knowledge is especially time-consuming for both the domain expert and the knowledge engineer.

BB1's blackboard control architecture ameliorates the knowledge acquisition bottleneck with generic knowledge sources for learning control heuristics. These knowledge sources exploit its blackboard representation of the current control plan, its explicit representation of knowledge sources, and its underlying model of the semantics of control. They generate new heuristics for use during an ongoing problem-solving session and new control knowledge sources to generate the heuristics during subsequent sessions. Some learning knowledge sources operate by interacting with a domain expert: others operate autonomously. (See [7] for more discussion.)

3. Building an Application System in BB1

A BB1 application system comprises domain and control blackboards. domain and control knowledge sources. and a basic control loop. Some of these are domain-dependent and specified by the system builder. Others are generic and provided by BB1. This section shows how to build an application system in BB1 with examples from the PROTEAN system. which solves the following problem:

> Given: a test protein's primary structure sequencing individual amino acids; its secondary structure locating individual alpha-helices, beta-sheets, and random coils within the primary structure; known atomic structures for individual amino acids: NOEs indicating proximate pairs of constituent atoms; vanderWaals radii indicating minimal distances between atom pairs: which atoms are on the surface of the protein; the general shape of the protein;
>
> Determine the (possibiy dynamic) locations of all constituent atoms in three-dimensional space.

3.1 Blackboard Specification

BB1's blackboard is a global database that records all solution elements generated during problem-solving. BB1 distinguishes domain and control blackboards. Each one comprises a number of different levels that group solution elements categorically. Solution elements recorded at each level are represented as objects with particular values on level -specific attributes.

The domain blackboard records solution elements for the domain problem. To define the domain blackboard, a system-builder specifies an ordered list of level names and, for each level, an ordered list of attribute names and initial values (if any). For example, Figure 1 shows the levels of PROTEAN's domain. blackboard. This blackboard's orthogonal axis represents three-dimensional spatial volumes. For illustration, Figure 2 shows the attributes specified for PROTEAN's Secondary-Anchor levei. Figure 3 shows an illustrative solution element representing the first alpha-helix in the test protein.

Secondary-Anchor	Pieces of secondary structure (alpha-helices, beta-sheets, random-coils) used to anchor partial solutions
Secondary	Pieces of secondary structure incorporated into partial solutions
Blob	Peptides, Side-Chains
Atomic	Atoms

Figure 1. PROTEAN's Domain Blackboard

Name	"Secondary" + number
Type	"Alpha-helix" or "Beta-sheet" or "Random-Coil"
Number	Type-specific sequential position
Sequence	Sequence of constituent amino acids
Number-AA	Number of constituent amino acids
Percent-AA	Percent of total amino acids
Internal-Constraints	NOEs among constituent atoms
External-Constraints	NOEs relating constituent atoms to other atoms
Constraints-To-Other-Structures	NOEs to atoms in each other piece of secondary structure
Parameters	Coordinates for three points in the structure to specify its size and spatial orientation

Figure 2. Attributes of Solution Elements at the Secondary-Anchor Level

Name	Secondary-Anchor2
Type	Alpha-Helix
Number	1
Sequence	(ASN5 TRP6 LEU8 ILE9 TRP10)
Number-AA	6
Percent-AA	3
Internal-Constraints	Nil
External-Constraints	5
Constraints-To-Other-Structures	((Secondary2 2 (5 6)) (Secondary3 0 Nil)) (Secondary4 2 (3 4)) (Secondary5 1 (2)))
Parameters	((Origin 0 0 0) (Reference 4 0 0) (Tip 0 0 14.0))

Figure 3. An Example Solution Element at the Secondary-Anchor Level

The control blackboard records solution elements for the control problem: what action should the system perform at each point in the problem-solving process? Figure 4 defines the domain-independent levels of the control blackboard and gives an example decision at each level. The control blackboard's orthogonal axis represents problem-solving time intervals measured in cycles. For illustration, Figure 5 defines the attributes used for the Focus level and Figure 6 shows an example Focusdecision.

Problem	Problem the system has decided to solve "Elucidate the structure of LAC-Repressor Headpiece"
Strategy	General sequential plan for solving the problem "Anchor all secondary structures: then refine all partial solutions that anchor at least one Secondary element"
Focus	Local (temporary) problem-solving heuristics "Anchor all secondary structures"
Policy	Global (permanent) problem-solving heuristics "Perform actions that generate control heuristics"
To-Do-Set	Pending problem-solving activities "Anchor-Helix helix 1 to Secondary-Anchor4 Anchor-Helix helix 1 to Secondary-Anchor5 Refine-Partial-Solution anchored by Secondary-Anchorl"
Chosen-Action	Problem-solving activities scheduled to execute "Anchor-Helix helix 1 to Secondary-Anchor5"

Figure 4. Levels of BB1's Control Blackboard and Summarized Examples

Name	"Focus" + number
Goal	Predicate or function to rate potential actions O-100
Criterion	Predicate to evaluate an expiration condition
Weight	Goal importance: O-l
Rationale	Reason for Goal
Creator	Action that created this decision
Source	Information that triggered the Creator
Type	Role in control plan, e.g.,"Strategic," "Solution-Based"
Status	Function in control plan, e.g., "Operative," "Inoperative"
First-Cycle	First operative cycle number
Last-Cycle	Last operative cycle number

Figure 5. Attributes Defined for the Focus Level

Name	Focus1
Goal	(Eq KS-Type 'Anchor)
Criterion	(for Element in ($Level 'Secondary-Anchor) always ($Find-One ((Level-Is 'Secondary)(Copies Element))))
Weight	8
Rationale	"Develop a comprehensive set of partial solutions before deciding which ones to refine at the blob level"
Creator	Chosen-Action 5
Source	Strategy1
Type	Strategic
Status	Operative
First-Cycle	6
Last-Cycle	20

Figure 6. An Example Solution Element at the Focus Level

3.2 Knowledge Sources

BB1's knowledge sources are interpretable, modifiable datastructureswiththeattributesdefined in Figure 7. Only a knowledge source whose Triggers and Pre-Conditions are true can execute its Action. The triggering, scheduling, and execution of knowledge sources are discussed in section 4.3. BB1 provides a simple menu-driven interface to facilitate knowledge source creation.

Name	Identifying string
Description	English-language summary of behavior
Condition =	
Trigger	Event-based predicates to evaluate KS relevance
Pre-Condition	State-based predicates to evaluate KS feasibility
Condition-Vars	Specification of condition variables to be used in action
Action =	
Rule(s) =	
LHS	Any predicates
RHS	Add/Modify blackboard nodes
KS-Features	From-BB, To-BB, From-Level, To-Level, KS-Type, Efficiency, Reliability, Author

Figure 7. Knowledge Source Attributes

Domain knowledge sources operate primarily on the domain blackboard and generate solution elements for the domain problem. Figure 8 shows an example PROTEAN domain knowledge source, Anchor-Helix, which incorporates a helix into a partial solution anchored by another piece of secondary structure. Anchor-Helix is triggered when a Secondary-Anchor is modified to indicate that it has at least two NOEs with a helix. Its action creates an element at the Secondary level that: (a) “copies” the Secondary-Anchor representing the heiix: (b) is “anchoredby” the triggering Secondary-Anchor; and (c) has Parameters that locate it relative to the triggering Secondary-Anchor. Its KS-Features provide descriptive information. such as the knowledge source’s efficiency and reliability, that might be useful to the scheduler.

```
Name              Anchor-Helix
Description       Locates a helix relative to another helix or a beta-sheet
                  with which it has at least two NOEs
Condition =
 Trigger           (Event-Level-Is Secondary-Anchor)
                   (Change-Type-Is Supersede)
                   (Changed-Attribute-Is Constraints-To-Other-Structures)
                   (Eq ($Value (Setq Newstructure (identify-Newstructure))
                               Type) 'Alpha-Helix)
                   (Greaterp (Setq Numberofnoes
                               (Count-Noes Event-Node Newstructure)) 1)
 Pre-Condition    T
Condition-Vars   (Newstructure Numberofnoes)
Action =
 Rule =
  LHS             T
  RHS              (Propose changetype ADO level Secondary
                    attributes
                    ((Relative-Location (Rel-Parms Event-Node Newstructure)))
                    links ((Copies Newstructure) (Anchoredby Event-Node)))
KS-Features       ((From-BB 'Domain) (To-BB 'Domain)
                    (From-Level 'Secondary-Anchor) (To-Level 'Secondary)
                    (Efficiency . 7) (Reliability (Itimes Numberofnoes 20))
                    (KS-Type 'Anchor) (Author "Barbara, Olivier, Mike")
```

Figure 8. Anchor-Helix: A PROTEAN Knowledge Source

Control knowledge sources operate primarily on the control blackboard and generate solution elements for the control problem. Figure 9 shows a generic control knowledge source, Implement-Strategy, which incrementally implements a Strategy decision as a series of prescribed Focus decisions. Implement-Strategy is triggered when either: (a) a new Strategy is posted on the blackboard; or (b) a previously posted strategic Focus decision's expiration criterion is met. In the first case, its action implements the "First-Focus" prescribed by the Strategy. In the second case, its action declares the expired Focus inoperative and: (a) if the expired Focus corresponds to the "Last-Focus" prescribed by the Strategy, it declares the Strategy inoperative; or (b) otherwise, it implements the next Focus prescribed by the Strategy. Its KS-Features provide descriptive information that might be useful to the scheduler.

```
Name              Implement-Strategy
Description       Incrementally implements a Strategy decision as
                     a series of prescribed Focus decisions
Condition =
 Trigger          (Or (And (Event-Level-Is Strategy)
                           (Change-Type-Is New)
                           (Setq Newstrategy Event-Node))
                      (And (Event-Level-Is Focus)
                           (Change-Type-Is Add)
                           (Setq Expiredfocus Event-Node)))
 Pre-Condition    (If Expiredfocus
                      Then (Eval ($Value Expiredfocus Criterion))
                           (Setq Oldstrategy
                                ($Find ((Sourceof Expiredfocus)))))
Condition-Vars  (Newstrategy Expiredfocus Oldstrategy)
Action =
 Rule1 =
  LHS             Newstrategy
  RHS             (Propose changetype Add level Focus
                   attributes ((Goal (Eval ($Value Newstrategy First-Focus)))
                     (Criterion (Eval ($Value Newstrategy Refocus-Criterion)))
                     (Weight ($Value Newstrategy Weight))
                     (Rationale ($Value Newstrategy Rationale))
                     (Type 'Strategic) (Source Newstrategy)
                     (Status 'Operative) (First-Cycle Next-Cycle))
 Rule2 =
  LHS             (Eq Expiredfocus (Eval ($Value Oldstrategy Last-Focus)))
  RHS             (Propose changetype Modify level Focus
                   attributes ((Status 'Inoperative))
                  (Propose changetype Modify level Strategy
                   attributes ((Status 'Inoperative))
 Rule3 =
  LHS             Expiredfocus
  RHS             (Propose changetype Modify element Expiredfocus
                   attributes ((Status 'Inoperative))
                  (Propose changetype Add level Focus
                   attributes ((Goal (Apply
                                        ($Value Oldstrategy Focus-Increment)
                                        ($Value Expiredfocus Goal)))
                     (Criterion (Eval ($Value Oldstrategy Refocus-Criterion)))
                     (Weight ($Value Oldstrategy Weight))
                     (Rationale ($Value Oldstrategy Rationale))
                     (Type 'Strategic) (Source Oldstrategy)
                     (Status 'Operative) (First-Cycle Next-Cycle))
KS-Features       ((From-BB 'Control) (To-BB 'Control) (From-Level 'Strategy)
                   (To-Level 'Focus) (Efficiency .9) (Reliability 100)
                   (Author "Barbara") (KS-Type 'Strategic))
```

Figure 9. Implement-Strategy: A Generic Control Knowledge Source

3.3 Basic Control Loop

BB1's basic control loop has three stages.

The first stage updates a To-Do-Set of pending knowledge source activation records (KSARs). each of which represents a unique combination of knowledge source and triggering events. This stage includes the following steps:

1. The most recently executed KSAR is removed from the To-Do-Set.
2. KSARs triggered by the executed KSAR's blackboard events are added to the To-Do-Set.
3. All triggered KSARs whose pre-conditions are true become "invocable."
4. All KSARs are rated against operative control heuristics on the control blackboard.
5. A priority is computed for each KSAR based on its ratings and an integration rule (e.g., sum of weighted weightings on operative heuristics) recorded on the control blackboard.
6. The user is invited to inspect the To-Do-Set, blackboard contents, or knowledge sources.

The second stage schedules a KSAR for execution, as follows:

1. An invocable KSAR in the current To-Do-Set is recommended, based on a scheduling rule (e.g., schedule the highest priority KSAR) recorded on the control blackboard.
2. The user is invited to approve, obtain an explanation of the recommendation, or override the recommendation with a preferred KSAR.
3. The scheduled KSAR's pre-conditions are re-evaluated and if any are no longer true, the KSAR is returned to the To-Do-Set and a new one is scheduled.

The third stage executes the scheduled KSAR.

4. An Illustrative BB1 /PROTEAN Trace

PROTEAN elucidates protein structure by reasoning at multiple levels of abstraction (e.g., secondary, blob, atomic) about the relative locations of structures (e.g., alpha-helices. amino acid side-chains, atoms) in a test protein. At each level, it generates multiple partial solutions. which are hypotheses regarding the relative locations of two or more structures. Within a partial solution. PROTEAN successively applies constraints to restrict the volume each constituent structure can occupy. These constraints include those provided in the problem description and additional constraints inferred from other hypothesized partial solutions. PROTEAN uses control heuristics to decide which partial solutions to expand and which constraints to apply. The problem is solved when PROTEAN has expanded consistent partial solutions that encompass the entire protein and satisfy all available constraints at all levels of abstraction.

As shown below, a user runs PROTEAN by instructing BB1 to load all PROTEAN files and to run the system on a specified data file, in this case the file Pseudo-Data. BB1 initiates PROTEAN problem-solving activity by executing a designated knowledge source, Post-The-Problem. It records this scheduling decision on the control blackboard in node Chosen-Actionl. Post-The-Problem retrieves data from the file Pseudo-Data and posts it on the blackboard in the node Problem1 .

```
What would you like to do: L
Load files for the system: PROTEAN
What would you like to do:R
Run PROTEAN on data in file: Pseudo-Data

I have created the event (Add Chosen-Actionl)
I have executed KSAR 1 Post-The-Problem
I have created the event (Add Probleml)
```

On Cycle 2 (shown below), BB1/PROTEAN[2] follows the user's requests to display the invocable KSARs in the current To-Do-Set, to display the blackboard node Probleml, and to recommend a KSAR for execution. At this point, there are no control heuristics posted on the control blackboard, so BB1/PROTEAN must select one of the three invocable KSARs according to a default LIFO principle. It records its recommendation of KSAR 2 in node Chosen-Action2 on the control blackboard and reports it to the user. When the user approves the recommendation, BB1 /PROTEAN executes KSAR 2, generating Policy2: (EQ To-BB 'Control), which influences all subsequent scheduling recommendations. (The node Policy1 is reserved to record the scheduling and integration rules.)

[2] For simplicity. the discussion does not distinguish BB1 and PROTEAN behaviors.

```
Cycle 2
To-Do-Set--T/I/A/N: I

Invocable KSARs:
KSAR 2 Reflect-Often   triggered by (Add Probleml)
KSAR 3 Post-Secondary-Anchors   triggered by (Add Probleml)
KSAR 4 Anchor-Then-Refine   triggered by (Add Probleml)

Display/Execute/Recommend/Charge-Ahead: D
Display Knowledge source/Blackboard/Level/Node: N
Node: Problem1

Problem1
Protein-Name           Pseudo-Protein
Primary-Structure      (ALA1 PHE2 TRY3 GLY4 ASN5 TRP6 LEU8 ILE9 TRPlO
                        GLYll GLN12 GLU13 TYR14 MET15 PRO16 VAL17
                        LEU18 ALA19 TRP20)
Secondary-Structure    ((R 1 ALA1 GLY4) (A 1 ASN5 TRP10) (R 2 GLYll GLN12)
                        (B 1 GLU13 PR016) (A 2 VAL17 TRP20))
NOEs                   ((1 ALA1 8 TRP20 12) (2 TYR3 12 ASN5 15)
                 ..     (3 TRP6 10 GLYll 7) (4 LEU8 10 GLN12 13)
                        (5 VAL7 10 VAL17 11) (6 ILE9 12 ALA19 9)
                        (7 TYR14 13 ALA19 19))

Display/Execute/Recommend/Charge-Ahead: R

I have created event (Add Chosen-Action2)
I recommend KSAR 2
OK/Why/KSAR#: OK

I am executing KSAR 2
I have created the event (Add Policy2)
```

On Cycle 3 (not shown), BB1/PROTEAN recommends KSAR 4, Anchor-Then-Refine triggered by the event (Add Probleml). because it satisfies Policy2. BB1 /PROTEAN executes KSAR 4, generating Strategy1 : Anchor then Refine.

On cycle 4 (not shown), BB1 /PROTEAN recommends KSAR 5, Implement-Strategy triggered by the event (Add Strategy1), because it satisfies Policy2. BB1/PROTEAN executes KSAR 5, implementing Strategy1's First-Focus: (Eq KS-Type ‘Anchor), which influences all subsequent scheduling recommendations.

On cycle 5 (not shown), BB1 /PROTEAN recommends KSAR 3, Post-Secondary-Anchors triggered by the event (Add Probleml), because: (a) it satisfies Policy2; and (b) neither it nor any other pending KSARs satisfies Focus1. BB1 /PROTEAN executes KSAR 3, generating five solution elements at the Secondary-Anchor level to represent each of the five pieces of secondary structure recorded in Problem 1.

On cycle 6 (shown below), BB1/PROTEAN recommends KSAR 6, Anchor-Helix triggered by the event (Modify Secondary-Anchor4) and records it in the node Chosen-Action6 on the control blackboard. 881 /PROTEAN explains the recommendation in terms of its control plan: its current scheduling and integration rules, and its operative Strategy, Focus, and Policy decisions. It also enumerates other KSARs with the same priority. The user overrides BB1 /PROTEAN's recommendation with KSAR 7, Anchor-Helix triggered by the event (Modify Secondary-Anchor5). **881 /PROTEAN** modifies the node Chosen-Action6, but preserves its own recommendation as well. BB1/PROTEAN executes KSAR 7, generating Secondaryl. which "copies" Secondary-Anchor2 and is "anchoredby" Secondary-Anchor5.

```
Cycle 6
To-Do-Set--T/I/A/N: I

Invocable KSARs:
KSAR 6 Anchor-He1 ix  triggered by (Modify Secondary-Anchor4)
KSAR 7 Anchor-He1 ix  triggered by (Modify Secondary-Anchor5)
[Other    KSARs]

Display/Execute/Recommend/Charge-Ahead: R

I have created event (Add Chosen-Action6)
. I recommend KSAR 6
OK/Why/KSAR#: W

KSAR 6 Anchor-He1 ix  triggered by (Modify Secondary-Anchor4)
Control Plan:
  Scheduling Rule: Highest Priority KSAR
  Integration Rule: Sum of Weighted Ratings
  Strategy1  Anchor-Then-Refine
    Rationale: Develop a comprehensive set of partial solutions before
               deciding which ones to refine at the blob level
    Focus1 (Eq KS-TYpe 'Anchor) Weight 8 Rating 100
  Policy2 (Eq To-BB 'Control) Weight 10  Rating 0
Priority: 800
KSARs with the same Priority:  KSAR 7 KSAR 8 KSAR 9

OK/Why/KSAR#: 7

I have created the event (Supersede Chosen-Action6)
I am executing KSAR 7
I have created the event (Add Secondaryl)
```

On Cycle 7 (shown below), **881 /PROTEAN** recommends KSAR 15, Understand-Preference triggered by the event (Supersede Chosen-Action7). 881 /PROTEAN explains its recommendation and executes KSAR 15, generating a program of activity to learn the heuristic underlying the user's preference for KSAR 7 over KSAR 6 on cycle 6. Briefly, Understand-Preference searches for a heuristic that captures the critical difference between the two KSARs underlying the user's preference. It operates by performing simple comparison operations on BB1's semantic representations of control information knowledge sources and solution elements and its syntactic representations of domain knowledge sources and solution elements. Understand-Preference also knows about data types (e.g., number, literal, list). corresponding comparative terms (e.g.. higher, value a versus *b,* different), and prototypical heuristic functions (e.g., Prefermore). By exploiting these several kinds of knowledge, Understand-Preference minimizes its reliance and demands upon the expert. Understand-Preference identifies the new heuristic:

```
(If (Event-Level-Is Secondary)
  Then (Prefermore
    (for X in ($Value Event-Node Constraints-To-Other-Structures)
       count (Greaterp (Elementno 3 X) 0)) 5)),
```

and records it as Policy3, which influences all subsequent scheduling decisions. It also creates a new control knowledge source, Maximize-Constraint, to generate the new heuristic during subsequent problem-solving sessions. (This section of the trace is explained in detail in [7].)

```
Cycle 7
To-Do-Set--T/I/A/N: I

Invocable KSARs:
KSAR 6 Anchor-He1 ix  triggered by (Modify Secondary-Anchor4)
[Other KSARs ]
KSAR 15 Understand-Preference triggered by (Supersede Chosen-Action6)

Display/Execute/Recommend/Charge-Ahead:  R

I have created the event (Add Chosen-Action7)
I recommend KSAR 15
OK/Why/KSAR#: W

KSAR 15 Understand-Preference  triggered by (Supersede Chosen-Action6)
Control Plan:
  Schedul ing Rule: Highest Priority KSAR
  Integration Rule: Sum of Weighted Ratings
  Strategy1  Anchor-Then-Refine
    Rationale: Develop a comprehensive set of partial solutions before
               deciding which ones to refine at the blob level
    Focus1 (Eq KS-TYpe 'Anchor) Weight 8 Rating 0
  Pol icy2 (Eq To-BB 'Control)  Weight 10 Rating 100
Priority: 1000

OK/Why/KSAR#: OK

I am executing KSAR 15

I would like to understand why you preferred KSAR 7 over KSAR 6.

Both KSARs involve the knowledge source Anchor-Helix.
Do you prefer KSAR 7 because of a difference in:
```

```
Event or Node: N
KSAR 6's Node is Secondary-Anchor4
KSAR 7's Node is Secondary-Anchor5
Do you prefer KSAR 7 because Secondary-Anchor5:
(A) has Type = Alpha-Helix, rather than Beta-Sheet?
(B) has a higher Number?
(C) has a different Sequence?
(D) has a different External-Constraints?
(E) has a higher Number-External-Constraints?
(F) has a different Constraints-To-Other-Structures?
(G) has a different Parameters?
Choose A/B/C/D/E: F

Do you prefer Secondary-Anchor5's Constraints-To-Other-Structures
because it:
(A) has more non-zero element3s?
(B) has a bigger range of element3s (O-2, rather than O-l)?
(C) has more non-null element4s?
(D) has a bigger range of lengths of element4s (O-2, rather than O-l)?
Choose A/B/C/D/E: A
What is the greatest possible number of non-zero element3s? 5
Why do you prefer more non-zero element3s? Can anchor more structures
How important is your preference (l-lo)? 8

I am recording Policy3:
   (Goal '(If (Event-Level-Is Secondary)
           Then (Prefermore (for X in
                 ($Value Event-Node Constraints-To-Other-Structures)
                 count (Greaterp (Elementno 3 X) 0)) 5)))
   (Rationale "Can anchor more structures")
   (Weight 8)

OK/No:OK

I am creating a new control knowledge source:
   Trigger: (Event-Level-Is Problem)
            (Change-Type-Is Add)
            (Eq ($Value Event-Node Type) 'Protein-Analysis)
   Pre-Condition: T
   Action:
      Rule1
         LHS T
         RHS (Propose changetype Add level Policy attributes
              (Goal '(If (Event-Level-Is Secondary)
                      Then (Prefermore (for element3 in
                        ($Value Event-Node Constraints-To-Other-Structures)
                         count (Greaterp (Elementno 3 X) 0)) 5))
              (Rationale "Can anchor more structures")
              (Weight 8)

OK/No:OK

What would you like to name the new KS: Maximize-Constraint
Please describe it: This KS generates a Policy favoring actions that
operate on structures that have NOEs to many other structures.
```

On cycle 8, shown below, BB1/PROTEAN begins autonomous problem-solving activities and continues until the user's deadline, cycle 25, whereupon it returns to interactive mode.

```
Cycle 8
To-Do-Set--T/I/A/N: N

Display/Execute/Recommend/Charge-Ahead:   C
Charge-Ahead until: (Eq Thiscycle 25)

I have created the event (Add Schedule8)
I am executing KSAR 8
I have created the event (Add Secondary2)

[Autonomous  Problem-Solving]

I have created the event (Add Chosen-Action24)
I am executing KSAR 30
I have created the event (Add Blob17)

Cycle 25
To-Do-Set--T/I/-A/N: A

[Etc.]
```

5. Comparison with HEARSAY-Ill and AGE

Erman, London, and Fickas [3] developed HEARSAY-III, a uniform blackboard architecture with separate, user-defined. domain and control blackboards and a default low-level scheduler that simply schedules any pending KSAR. 881 shares HEARSAY-Ill's objectives and expands upon its functionality. Thus, while HEARSAY-III aims to support control problem-solving, it provides no theory of effective control problem-solving and no specific mechanism for implementing control problem-solving. By contrast, BB1's blackboard control architecture embodies a theory of effective control behavior (see [6]) and provides a well-defined blackboard structure and mechanism for achieving that behavior. These theoretical underpinnings also provide the foundation for BB1's explanation and learning capabilities.

Nii and Aiello [8] developed AGE, a more constrained blackboard architecture whose objectives differ from BB1's objectives. First, 881 aims for explicit, dynamic control, whereas AGE aims for efficient control. AGE uses a labelling scheme to streamline knowledge source invocation. Knowledge source conditions specify tokens that correspond to pre-determined event labels. Conversely, knowledge source actions assign labels to the blackboard events they create. AGE's scheduler sequentially selects blackboard events and, for each one, executes a pre-determined sequence of knowledge sources whose conditions specify its assigned token. This scheme is more efficient, but less flexible and powerful than BB1's dynamic control planning. Second, AGE and 881 provide complementary kinds of explanation. AGE explains solution elements for the domain problem by reviewing the sequence of knowledge source actions that produced them. 881 explains problem-solving actions by showing their roles in the dynamic control plan. Ongoing efforts to expand BB1's explanation capabilities may incorporate AGE's kind of explanation. Third, AGE and 881 provide different system-development support. AGE provides an elaborate interface for creating and editing knowledge sources, the blackboard structure, and a variety of supporting functions, whereas 881 provides only an interface for creating knowledge sources. Finally, AGE has none of BB1's learning capabilities.

6. Costs and Benefits of Using BB1

BB1 provides a structure and mechanism for building AI systems that perform explicit, dynamic control planning concurrently with efforts to solve particular domain problems. It permits extreme flexibility in system behavior, ranging from rigorously strategic behavior to unbridled opportunism. It permits systems to develop and behave in accordance with complex control plans whose operative strategies. heuristics. and integration and scheduling rules can change repeatedly in the course of problem-solving. As a consequence, systems built in BE31 can improve the efficiency of search by adapting to a wide range of unanticipated problem-solving situations.

BB1 provides a foundation and mechanism for detailed explanation of its problem-solving behavior It shows how specific actions fit into its dynamic control plan, which of its operative heuristics favor or disfavor those actions, what priorities the actions have, and which other actions have the same priorities. As a consequence, the behavior of systems built in BB1 is easy to understand.

BB1 also provides a foundation and knowledge sources for learning control heuristics. Some learning knowledge sources obviate the knowledge engineer and relieve the domain expert. Others obviate both the knowledge engineer and the domain expert. As a consequence, systems built in BB1 improve their own behavior and ameliorate the knowledge acquisition bottleneck.

BB1 also provides a flexible run-time user interface enabling a user to inspect its data structures, observe its behavior, obtain explanation of scheduling recommendations, override scheduling recommendations, and instruct it to operate autonomously until specified conditions occur.

BB1 entails high computational and storage costs, making it impractical for high-performance application systems. However, it offers a modular environment for experimental development of application systems. System builders can introduce, modify, and remove control knowledge sources independently. They can exploit previous system development efforts by transferring and combining control knowledge sources from different systems. They can use BB1's learning knowledge sources to elicit control heuristics from domain experts and code them in appropriate BB1 representations. After configuring and refining an appropriate set of control knowledge sources, the system builder can compile the control plan they entail in a more efficient, run-time control procedure.

References

[1] Buchanan, B., Jardetzky, O., Lichtarge, O., Hayes-Roth, B. Altman, R., and Hewett, M.
PROTEAN.
Technical Report, Stanford, Ca.: Stanford University, 1984.

[2] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R.
The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty.
Computing Surveys 12:213-253, 1980.

[3] Erman, L.D., London, P.E., and Fickas, S.F.
The design and an example use of Hearsay-Ill.
Proceedings of the Seventh International Joint Conference on Artificial Intelligence :409-415, 1982.

[4] Goldin, S. E., and Hayes-Roth, B.
Individual differences in planning processes.
Technical Report N- 1488-ONR, Santa Monica, Ca.: Rand Corporation, 1980.

[5] Hayes-Roth, B.
Flexibility in executive processes.
Technical Report N- 1170-ONR, Santa Monica, Ca.: Rand Corporation, 1980.

[6] Hayes-Roth, B.
A blackboard architecture for control.
Artificial Intelligence Journal in press, 1985.

[7] Hayes-Roth, B., and Hewett, M.
Learning Control Heuristics in a Blackboard Environment.
Technical Report, Stanford, Ca.: Stanford University, 1984.

[8] Nii, H.P., and Aiello, N.
AGE (attempt to generalize): A knowledge-based program for building knowledge-based programs.
Proceedings of the International Joint Conference on Artificial Intelligence 6:645-655, 1979.