Report 85-24
Stanford -- KSL

Scientific DataLink

Research on Blackboard Architectures at the Heuristic Programming Project.
H. Penny Nii,
May 1985

Knowledge Systems Laboratory **May 1985**

Report No. KSL-85-24

Research on Blackboard Architectures at the Heuristic Programming Project

by

H. Penny Nii

Stanford Knowledge Systems Laboratory
(incorporating the Heuristic Programming Project)
Stanford University
701 Welch Road (Building C)
Palo Alto, CA 94304

Research on Blackboard Architectures at the Heuristic Programming Project[1]

H. Penny Nii
Heuristic Programming Project
Computer Science Department
Stanford University
701 Welch Road, Building C
Palo Alto, California 94304

Abstract

For the past decade the Heuristic Programming Project (HPP) at Stanford University has been involved in research, development, and application of Blackboard frameworks for problem solving and hypothesis formation. The Blackboard framework was originally conceived at Carnegie-Mellon University during the Speech Understanding Project [13] in the early 1970s. Researchers at HPP have contributed significantly to the scientific and practical development of the Blackboard framework. The next project after CMU's HEARSAY to use the framework was the HASP [15] system for passive sonar signal understanding. Subsequent efforts involved experiments with scientific applications (x-ray crystallography [5]), intelligence problems (ELINT and COMINT), and planning [11], as well as the development of the first software tool to assist knowledge engineers in constructing systems using the Blackboard framework (AGE-1 [1]).

During the last decade, the Blackboard framework has proven to be a very flexible and powerful software concept for organizing the formulation, implementation, and processing of knowledge-based systems. We are now entering the second decade of research in the Blackboard problem solving framework, with focus at HPP in the following areas: (1) extensions of the basic concepts implemented in AGE-1, to address, for example, reasoning with uncertain data (which to date has received only ad-hoc treatment); (2) a new architecture and development environment, BB1, that implements methods for explicitly controlling the reasoning; and (3) the design of and experimentation with multiprocessor architectures using the Blackboard as an organizing framework. This paper summarizes these three efforts.

---

[1]This is a summary of three projects. The project leaders are Barbara Hayes-Roth for BB1, H. Penny Nii for AGE, and Edward A. Feigenbaum for Multiprocessor Architectures. These projects are made possible by grants from DARPA, NASA, and Boeing.

Background

The foundation for a knowledge-based system is the problem solving framework in which an application is implemented. Some of the more popular alternatives used to build knowledge systems are production systems, backward-chained reasoning, logic programming, heuristic search, and the Blackboard framework.

Many of the applications implemented in production systems have been written in the OPS language [8]. In this framework, knowledge is represented as a set of homogeneous rules that are scanned for applicability in a data base that contains the current state of solution. Backward chaining also has a homogeneous set of rules, but the search for applicable rules is driven by a hierarchy of goals and sub-goals. The best known system for implementing this type of program is EMYCIN [4]. In logic programming, represented by the PROLOG [3] language, both knowledge and data are represented homogeneously as logical assertions. Problem solving in this system is accomplished by proving a theorem that represents a solution goal. Heuristic search generates solutions either by algorithms or by heuristic knowledge, and uses additional heuristic knowledge to guide its search into the most promising areas of the solution space.

One of the most powerful knowledge-based system frameworks is the *Blackboard framework*. In this framework many different types of knowledge sources jointly contribute to the problem solution by forward, backward, or opportunistic reasoning via a common solution memory called the *blackboard*. In the Blackboard framework, knowledge can be represented in a variety of ways, for example, as production rules, objects, tables, or procedures. The evolving solution state is partitioned into hierarchically organized analysis levels, and the knowledge sources are activated whenever they can contribute toward a solution. The power of the Blackboard framework lies in its organization of knowledge and the solution state, the evolving nature of the solution, its ability to handle incomplete and errorful data, the use of explicit control knowledge (i.e., meta-knowledge) to guide the application of domain-specific problem solving knowledge, and the flexibility that allows the use of various knowledge representation schemes and inference mechanisms as appropriate.

Blackboard Framework for Problem Solving

In its most abstract and simplest form, the Blackboard framework (see, for example, [6], [12], and [14]) can be described as follows:

1. There are diverse *knowledge sources* that are kept separate and independent;
2. There is a global data base, the *blackboard*, that is used as a means of communication and interaction among the knowledge sources; and
3. The knowledge sources *respond to changes* in the blackboard.

In the Blackboard framework the application task is partitioned along three major dimensions. First, the knowledge relevant to solving the whole problem or parts of the problem is partitioned into various specialized modules (i.e., knowledge sources) that are independent of each other. In the HASP program, for example, knowledge about the noise-producing parts of ships is separate and independent from knowledge about behavior of acoustic signals in the ocean. The existence of one knowledge source does not depend on the existence of the other, and they are conjointly useful only within the context of a particular task to be performed.

The second partitioning is in the blackboard, which holds the intermediate, evolving solutions and through which the knowledge sources communicate. The most common partitioning is a hierarchical decomposition of the solution space into intermediate solution levels where knowledge sources can transform information on one level to information on another level. In HASP some of the levels of analysis are platforms, sources, and harmonic sets; some of the knowledge sources form harmonic sets, classify sources, and track platforms. Of course, the blackboard can be partitioned in many other ways; for example, along the spatial or temporal dimension.

The third partitioning occurs in the variety of ways in which knowledge can be manipulated and applied, commonly called the 'knowledge utilization strategy.' A blackboard program can dynamically decide when and in what manner to use the available knowledge sources, based on how the solution is unfolding. For example, in HASP, the system might process all information related to the target-of-interest before processing information related to other types of platforms. The specific utilization and management of knowledge sources are highly dependent on the application itself and on how the control mechanisms are actually implemented.

Given this general description of the Blackboard framework, a software system implementing the framework can be designed in many ways. For example, HEARSAY-II and HASP have significant differences in the way knowledge sources are represented as well as in the way they are activated (see [14] for some comparisons). BB1 and AGE-1 are software tools for building Blackboard systems that also reflect some basic differences in expressing and using control. For example, BB1 has two blackboards, one to hold data and the emerging solutions and one to hold control-related information. The knowledge is divided into domain and control knowledge and these operate on their respective blackboards. In multiprocessor Blackboard systems the concept of a globally accessible blackboard differs dramatically depending on whether the multiprocessor systems has shared memory or distributed memory. In sequential systems, control is concerned with the best serial application of knowledge at an given point in the problem-solving process; the same concern represents a potential bottleneck in multi-processor systems. Until recently, little research has been done on the implications of various degrees of control, whether they be in sequential or multiprocessor environment. A

Blackboard architecture provides a useful medium in which this research can be conducted, because all information, including control information, can be made available on the blackboard. Control in Blackboard systems is now a major topic of research.

AGE and Extensions to Basic Concepts

AGE-1 is a software tool that was developed between 1977 and 1982, during the same time that the EMYCIN [4] and UNITS [18] software tools were being developed at HPP. EMYCIN, which uses backward-chained reasoning, was derived directly from the MYCIN [2] application program by deleting MYCIN's medical knowledge. UNITS, an object-oriented knowledge representation system, was developed for the MOLGEN [9] program. AGE-1, however, was developed as a general software tool for building application programs using a Blackboard framework. It contains an extensive environment for specifying, editing, and debugging the program. The core of the system is simple in design and consists of the following components:

1. The global data base (the blackboard) in which emerging solutions are posted. The elements on the blackboard are organized into levels and represented as a set of attribute-value pairs (a frame).
2. Globally accessible lists on which control information is posted (e.g. lists of events, expectations, etc.).
3. An indefinite number of knowledge sources, each consisting of indefinite number of production rules.
4. Various kinds of control information that determine which blackboard element to focus on and what knowledge source to use at any given point in the problem solving process.

During execution of a user program, AGE-1 provides a simple and fixed control loop:

1. A focus-of-attention is chosen by selecting one event, expectation, or goal.
2. A knowledge source relevant to the focus-of-attention is selected and executed.
3. The action of a rule inside the executing knowledge source can
    a. make changes to a blackboard element's attribute values or add a blackboard element at a particular level (post event),
    b. indicate that some change to a blackboard element is expected in the future (post expectation), or
    c. indicate that some change to a blackboard element is desired (post goal).

This relatively simple architecture has been tested by various applications at HPP and elsewhere, and indications are that it is sufficient for a wide variety of applications. However, as we move on to the next generation of Blackboard systems, several important and interesting issues remain to be studied; two of them are listed here. First, there may be multiple uses of the global blackboard by different human or computer

agents. One can build a symbiotic system in which both programmed knowledge sources and users can contribute to the evolving solution on the blackboard. At minimum, such a system needs a language that supports user interaction with the blackboard and mechanisms that maintain coherency while integrating actions by users with those by knowledge sources. Second, reasoning from uncertain evidence, of special importance in signal-understanding tasks, has received only ad-hoc treatment to date. Aside from the general difficulty in dealing with data collected over time, the major reason for this ad-hocness is that the calculus of determining confidence is primarily application dependent. Confidence in the accuracy of signal data depends on the type of signal, the signal detection device, the signal processing algorithm, the noise level of the environment, and so on. Under these circumstances, no single algorithm suffices. One way to view this problem is as a subset problem of the overall application task requiring knowledge about how to handle uncertainty. A possible approach that integrates this problem with assessment problem is given below.

Of special interest is the use of the Blackboard framework to mount assessment problems. The Blackboard framework can be thought of as a framework for information fusion. Whether the information is available in the form of signals or of symbols, various sources of knowledge contribute to make a coherent story from the various sources and forms if information. The medium of fusion is the blackboard data structure, and this data structure also represents a coherent interpretation of the data.

Assessment is an interpretation of the information on the blackboard, which in turn is an interpretation of external data. Assessment itself is a goal-directed activity, in that the interpretation of the information on the blackboard depends on what the viewer is looking for. For example, given a blackboard containing a battlefield scene, a person in charge of supplies will assess the situation differently from a person who has to capture an enemy position.

Assignment of evidential and inferential weights to information on the blackboard can be viewed as localized assessment. In current systems, some arbitrary weight is assigned to incoming data based on its reliability, in the same way that inference is given an arbitrary weight based on some sense of confidence in the knowledge. In a system that has disparate sources of data and knowledge and in a situational hypothesis that is often evolving over time, the traditional confidence factor calculations [17] are inadequate or even meaningless. In many applications how much weight, or credibility, a piece of information should carry depends on the context in which the information is found as well as on the credibility of its source, whether that source be external or the result of some inference. The calculation of the credibility of information under such circumstances is probably best evaluated heuristically, much in the same way that the whole blackboard is assessed heuristically within a given context and a goal.

One possible design for an assessment system is to use the 'fusion blackboard' as one of the inputs. The system would most likely have other inputs and a set of knowledge sources specific to the assessment goal, as well as its own 'assessment blackboard,' on which to post an ongoing assessment. Neither this type of multi-tiered (and potentially distributed) blackboard system nor an interactive blackboard system can be built using the current AGE-1 system. Adding these capabilities would require a redesign that might change some basic constructs in major ways and might also require extensions to the current definition of the Blackboard system. Because most of the issues to be addressed are problem dependent, the best we can hope for in generalizing the solution is to work with a class of applications with similar characteristics. We are currently reviewing designs for a new tool that will be more appropriate than AGE-1 for the purposes described above.

BB1: Blackboard Control Architecture

Usually, building complex application programs has meant building complex control structures. This is a natural consequence of a strategy of 'divide-and-conquer'. Having broken a problem into manageable subproblems, we must determine how and when to bring the sub-problems together. In addition, the quality of knowledge that can be brought to bear at different points in the problem solving process will vary. For example, if there is not much situation-specific knowledge to be applied at a particular point, a system can resort to a method of generating possible solutions and testing them for validity. Thus, the control problem is which of its potential actions a program should perform at each point in the problem solving process. BB1 [10] is a relatively new project whose goal is to explore the answers to this question.

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 is used. It 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's control enables programs to achieve the following behavioral goals:

1. Make explicit control decisions that determine which problem solving actions they perform at each point in the problem solving process.
2. Decide what actions to perform by reconciling independent decisions about actions they should perform and actions they can perform.
3. Adopt variable grain-size control heuristics, including global strategies (e.g., in the PROTEAN project, described below, first anchor all pieces of secondary structure in partial solution; 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. 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. Adopt, retain, and discard individual control heuristics in response to dynamic problem solving situations.

6. Decide how to integrate multiple control heuristics of varying importance.

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

8. 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 wide range of unanticipated situations.

The BB1 architecture is being tested and refined through its use in the PROTEAN project. PROTEAN is an application program whose task is to elucidate protein structures by reasoning at multiple levels of abstractions (e.g. 'secondary,' 'blob,' atomic) about the relative locations of substructures (e.g. alpha-helices, amino acids, 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 substructures. Within a partial solution, PROTEAN successively applies constraints to restrict the volume each constituent substructure can occupy. These constraints include those provided in the problem description plus additional ones 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 abstractions.

Multiprocessor Architectures

The basic problem attacked by our research in multiprocessor architectures is the use of parallelism to speed up the run time of knowledge-based systems. Projected performance limits of uniprocessors indicate that parallelism must be used to attain the performance needed for most significant problems (3 to 4 orders of magnitude increase [7]). Our work is directed toward the support of parallelism at all levels of the system -- the application level, the problem solving framework level, the knowledge representation level, the programming language level, and the machine level. At each level we will be looking for a small factor (2 - 10) in speedup through parallelism. Our assumption is that gains at each level of computational hierarchy will reinforce each other multiplicatively (see, for example, [16]) as we go from the application level to the machine level primitives.

Instead of attacking the problem in a general way, we chose to focus on one class of applications (information

fusion) and one problem solving framework (the Blackboard). The Blackboard framework is well suited to exploit concurrency at various levels of processor granularity. Moreover, at least for the class of applications under consideration, the framework also admits the use of 'high-level' pipelined processing where appropriate.

With a very coarse grain size -- for example, a few to tens of powerful processors communicating via a local area network, where each processor handles a large partition of a problem -- a Blackboard system can be decomposed into multiple blackboards and sub-blackboards. The form of this decomposition depends on the particular application. For example, many situation understanding applications require the analysis of diverse streams of collected data (e.g., active and passive radar data, sonar data, intercepted communications data, and intelligence report data) followed by a fusion of the results of these analyses into a single, consistent analysis of the situation. Each of the analysis systems as well as the fusion system could be realized as an independent blackboard system, implemented on a separate processor. Since the communication between the systems consists primarily of analysis reports and fusion feedback, only relatively low bandwidth communication channels are required.

At a less coarse level of processor granularity, each of the blackboards in such a system could be partitioned into sub-blackboards along various dimensions, for example, spatial or frequency. In the HASP program, for example, the ocean can be partitioned into grids each with its own sub-blackboard representing the situation in a smaller piece of the ocean. For such partitioning, there will be trade-offs between the number of partitions, the communication bandwidths, and the sharing or replicating of knowledge sources. These trade-offs need to be investigated.

The knowledge sources in the Blackboard framework are operationally independent. At a finer level of granularity -- for example, hundreds of processors -- each knowledge source could be associated with one or more processors for concurrent execution. In order to understand the speedup potential of such associations, we need to investigate the relationships between knowledge sources and the 'areas' of the blackboard that they 'touch,' the penalties involved in replicating portions of the blackboard, and the inherently sequential (or nonsequential) nature of the reasoning processes used.

At a yet finer level of processor and memory granularity -- for example, thousands of processors -- the individual entities on the blackboard could be considered as active objects (i.e., able to independently respond to messages) and, as such, could be assigned to distinct processors organized as some type of network. With this type of blackboard-entity-to-processor association, the entities could, for example, concurrently notice changes in themselves and 'trigger' knowledge sources as well as respond directly to search queries.

At present there is little or no information regarding (a) the qualitative aspects of programming a knowledge-based system within a multiprocessing setting or (b) the potential quantitative gain resulting from a multiprocessor-based knowledge system. To gain the best understanding of the effectiveness of a parallel implementation of a knowledge-based system on multiprocessors, it is necessary to study the programming problem and the performance issues all the way from the problem specification down to the detailed interactions of particular features of various multiprocessor architectures. To this end, a parameterized architectural simulator is being built to test concurrency at all levels from the application formulation to a multiprocessor system architecture.

Summary

In the long term, research on Blackboard systems and environments for building them will produce new tools for constructing more powerful expert systems, but some important issues remain to be studied. We are currently studying new architectures that can solve more complex problems in some systematic way, that can reason about their own control strategy, and that can run concurrently in multiprocessor environment.

References

[1] Aiello, N., C. Bock, H. P. Nii, and W. C. White.
Joy of AGE-ing
Heuristic Programming Project, Stanford University, Stanford Ca. 94305, 1981.

[2] Buchanan, Bruce G. and Edward H. Shortliffe.
Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.
Addison-Wesley, Reading, MA, 1984.

[3] Clocksin W.F. and C.S. Mellish.
Programming in PROLOG.
Springer-Verlag, Berlin, 1981.

[4] van Melle, W., A. C. Scott, J. S. Bennett, M. Peairs.
The EMYCIN Manual
Heuristic Programming Project, Stanford University, Stanford Ca. 94305, 1980.

[5] Engelmore, R. S. and A. Terry.
Structure and Function of the CRYSALIS System.
Proc. of IJCAI 6 :251 - 256, 1979.

[6] Erman, L.D., F. Hayes-Roth, V. R. Lesser, D. Raj Reddy.
The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty.
ACM Computing Survey 12:213 - 253, June, 1980.

[7] Feigenbaum, Edward A., Harold Brown, Bruce Delagi, Richard Gabriel, Penny Nii, Thomas Rindfeisch.
Advanced Architectures Project: Scope and Approach.
Technical Report HPP-84-40 (Working Paper), Stanford Heuristic Programmning Project, October, 1984.

[8] Forgy, C. and J. McDermott.
OPS, A Domain-Independent Production System Language.
Proc. IJCAI-77 1:933-939, 1977.

[9] Friedland, Peter.
MOLGEN: Applications of Artificial Intelligence and Symbolic Computations to Molecular Biology.
Proceedings of the First Annual Battelle Conference on Genetic Engineering 4:171 - 182, April, 1981.

[10] Hayes-Roth, Barbara.
BB1: An Architecture for Blackboard Systems That Control, Explain, and Learn about Their Own Behavior.
Technical Report HPP-84-16, Stanford Heuristic Programming Project, December, 1984.

[11] Hayes-Roth, B. and F. Hayes-Roth.
Modelling Planning as an Incremental, Opportunistic Process.
Proc. of IJCAI 6 :375-383, 1979.

[12] Lesser, Victor R. and Daniel D. Corkill.
The Distributed Vehicle Monitoring Testbed: A Tool for Investigation Distributed Problem Solving Networks.
The AI Magazine Fall:15 - 33, 1983.

[13] Newell A., J. Barnett, C. Green, D. Klatt, J. C. R. Licklider, J. Munson, R. Reddy, and W. Woods.
Speech Understanding System: A Final Report of a Study Group.
North-Holland, 1973.

[14] Nii, H. Penny.
An Introduction to Knowledge Engineering, Blackboard Model, and AGE.
Technical Report HPP-80-20, Heuristic Programming Project, C. S. Dept., Stanford University, March, 1980.

[15] Nii, H. P., E. A. Feignebaum, J. J. Anton, and A. J. Rockmore.
Signal-to-Symbol Transformation: HASP/SIAP Case Study.
AI Magazine 3:2:23 - 35, 1982.

[16] Reddy, Raj and Allen Newell.
Multiplicative Speedup of Systems.
In Anita Jones (editor), *Perspectives on Computer Science*, . Academic Press, New York, 1977.

[17] Shortliffe, Edward H.
Computer-Based Medical Consultation: MYCIN.
American Elsevier, New York, 1976.

[18] Smith, Reid and Peter Friedland.
Unit Package User's Guide
Heuristic Programming Project, Stanford University, Stanford Ca. 94305, 1980.