(12)

Heuristic Programming Project
October 1982 - September 1985
Final Report

Principal Investigators Edward A. Feigenbaum and Bruce G. Buchanan

March 4, 1986

Sponsored by Defense Advanced Research Projects Agency (DoD)
ARPA Order No. 3423
Monitored by Space and Naval Warfare Systems Command
Under Contract No. N00039-83-C-0136

Table of Contents

Heuristic Programming Project
October 1982 - September 1985
Final Report

1. Introduction

This report summarizes the research performed at Stanford University under ARPA Order No. 3423, Contract N00039-83-C-0136. The overall purpose of the contract was basic and applied research into the science and engineering of knowledge-based systems. This work focused on five important and interrelated problems in building and using Knowledge-based Systems:

1. Basic Research in Knowledge-based Systems Design and Implementation
2. Intelligent Agents and tools for interacting with complex systems, planning, and cooperative problem solving
3. The Blackboard Model and integration of knowledge from different sources
4. Advanced Knowledge-based System Architectures for high performance parallel processing hardware and software
5. Knowledge-based VLSI tools for design and testing of integrated circuits

All of the research described in this report was performed within the Knowledge Systems Laboratory (KSL), formerly known as the Heuristic Programming Project (HPP). Professors Edward Feigenbaum and Bruce Buchanan were the Principal Investigators of the contract, and Mr. Thomas C. Rindfleisch is the Director of the KSL.

Much of the research in this contract was performed on high performance LISP workstations, many of them purchased under this contract. A DEC VAX 11/780 facility purchased under a previous contract from ARPA with the HPP (contract # MDA903-80-C-0107) was used to support basic research into heuristic methods as well as the VLSI Theory work. Additional research was performed using the facilities of the NIH-sponsored SUMEX-AIM computer system (a Digital Equipment Corporation 2060.)

Sections 3 through 7 of this report summarize the KSL research under this contract. Publications resulting from this work contain the details of results and methodology, and are listed at the end of each section. Section 8 describes the computing environment of the KSL, the equipment purchases made under this contract, and the systems research and development that support the computing resources of the KSL.

2. Knowledge Systems Laboratory Organization

The Knowledge Systems Laboratory is a professional team of professors, research scientists, programmers, and research assistants within the Stanford University Computer Science Department. It has long-standing research goals of understanding the representation, use, and acquisition of knowledge for high performance problem solving by knowledge-based systems. Its research paradigm is experimental: apply the methodologies of applied artificial intelligence to difficult problems in science, engineering, and medicine in order to test the strengths and limitations of current methods and drive the development of improvements. The KSL reflects the complexity and diversity of the collaborating yet distinct research programs, all of which formerly were collected under the single project known as the HPP. The KSL is a laboratory with five groups, each broadly concerned with experimental research on artificial intelligence.

- *Heuristic Programming Project (HPP)*, Edward A. Feigenbaum, Scientific Director

- blackboard systems, concurrent system architectures for AI, and the modeling of discovery processes.

- *The HELIX Group*, Bruce G. Buchanan, Scientific Director - machine learning, transfer of expertise, and problem solving.
- *The Medical Computer Science Group (MCS)*, Edward H. Shortliffe, Scientific Director - research on and advanced application of AI to medical problems.
- *The Logic Group*, Michael R. Genesereth, Scientific Director - formal reasoning and introspective systems.
- *The Symbolic Systems Resources Groups (SSRG)*, Thomas C. Rindfleisch, Scientific Director - research on and operation of computing resources for AI research, including the SUMEX facility.

The KSL provides an environment that encourages the continued close communication, collaboration, and sharing of resources that have characterized the work of the HPP in the past. KSL work ranges from basic research in the fundamentals of knowledge representation and acquisition to specific projects that produce functional problem-solving systems in areas as diverse as cancer therapy, protein structure determination, and design of integrated circuits.

In the context of this ARPA contract, KSL work was funded in the following areas:

- Research on the Acquisition of Knowledge
- Software Engineering Tools for Knowledge-based Systems
- Reasoning about Mechanism and Design
- Elucidating the Nature of Heuristics
- Implementation of Intelligent Agents for user interface, planning, and problem solving
- Research on the Blackboard Model to improve the utility, identify principles and limitations, and to understand the behavior and performance of the model
- Software and Hardware Architectures for Advanced Knowledge-based Systems

For each of these areas, the resulting work during the course of the contract will be described along with references to the relevant HPP publications. A complete bibliography of KSL publications for the period of the contract is given at the end of this section.

3. Basic Research in Knowledge-based System Design

3.1. Research on the Acquisition of Knowledge

Our basic research in knowledge acquisition has followed several approaches, including knowledge engineering as well as learning from examples, by analogy, by watching an expert, by chunking, by discovery, and from written text.

Knowledge Engineering. To establish a problem-solving framework and establish the terminology, ROGET carries on a dialogue with an expert to formulate the working vocabulary and organization for the knowledge base. ROGET is an EMYCIN-based expert system whose domain is knowledge engineering. ROGET is discussed in HPP-83-24[1].

Two interactive systems are being developed to analyze the contents of an emerging knowledge base. MARCK learns control knowledge from PROTEAN[2] by examining the differences between the recommendations of the system and of an expert running PROTEAN. MARCK suggests new guidance heuristics that will guide choices in similar situations in the future. It is described in KSL-85-2[3].

OPAL, a knowledge engineering tool, aids in the definition of new knowledge in ONCOCIN, an advisor for cancer therapy management. OPAL has considerable knowledge about some cancer types and treatment plans, and uses interactive graphics to elicit information from a specialist.

Induction. PRE implements a method for interpreting the observed data, using "theory-driven data interpretation" that propagates constraints to determine a consistent interpretation of the data. RL is a system that learns from solved cases, using knowledge about how to learn from examples to build rules and develop theories that provide a consistent interpretation of the data. PRE and RL are discussed in HPP-84-17[4], HPP-84-46[5], and KSL-85-44[6].

Analogic reasoning is an important method for constructing knowledge bases, using one base as a stepping stone for the construction of analogous sets of knowledge. Our research with NLAG has focused on the appropriate use of analogies in the domain of electrical circuits. NLAG exploits analogies between fluid flow and electrical concepts such as voltage and current and saves those relations that help solve problems in the new domain. See KSL-HPP-84-8[7] for more information.

Learning by watching is a form of learning from examples, observing the problem-solving behavior of a specialist much as an apprentice would watch an expert. ODYSSEUS has several stages: inducing the rules and frame knowledge for a system, inferring how the specialist reasons, and acquiring missing knowledge by asking specific questions of the expert. ODYSSEUS is used in the HERACLES system (a generalized version of NEOMYCIN) to assist in expertise transfer by inferring a model of the behavior of an expert. In GUIDON2, ODYSSEUS attempts to infer the model of a novice's behavior in learning. Further information about ODYSSEUS is found in HPP-84-29[8] and KSL-85-26[9].

Chunking. The SOAR project uses a somewhat different approach to knowledge acquisition. In problem solving we often find a way to combine several steps into a single operation or "chunk", improving performance and efficiency. A chunk can then be represented as a rule that identifies which task aspects were useful in reaching the goal. SOAR is a general problem-solving system that can use chunking to improve all aspects of its performance. See HPP-84-39[10] and KSL-85-34[11].

Learning from Literature. We have implemented a prototype system of a knowledge acquisition program to obtain information from published articles. REFEREE interacts with an informed (though nonexpert) reader to acquire text, critiquing the design and execution of a reported clinical trial on a drug, for example. REFEREE then infers how well the subjects were randomized and presents a justification of why someone should or should not believe the conclusions of the paper. For further information, see HPP-84-49[12].

In all of the above areas, prototype systems have been completed. Active research continues on these and other approaches to knowledge acquisition. For more information on the research of the KSL in knowledge acquisition, see KSL-85-38[13].

The research described above has been funded in part by the following contracts and grants: DARPA N00039-83-C-0136; NIH RR 00785 from the Bio medical Research Technology Program, Division of Research Resources; ONR N00014-79-C-0302; NSF IST 83-12148; and Boeing W266875.

3.2. Software Engineering Tools

3.2.1. GLISP: Representation-Independent Programming

GLISP is a large package built on InterLisp that provides a conversational environment in which object-oriented systems can be constructed. It manages context and provides a graphical interface that frees the knowledge system programmer from many low-level programming concerns. GLISP is compiled into LISP using a knowledge base of object descriptions, treating LISP objects and objects in AI representation languages uniformly. This makes program code independent of the data representation used, and permits changes of representation without changing code. Abstract data types and inheritance of properties and behavior from multiple superclasses are supported.

A GLISP Display Inspector/Editor, GEV, was developed as a knowledge-based interactive editor for use on a Lisp machine with an high-resolution display. GEV uses GLISP structure descriptions to interpret and intelligently display LISP data, allowing the user to select objects with a mouse and focus attention on objects of interest.

GLISP was built by Professor Gordon Novak of the University of Texas while a visiting faculty member at Stanford. GLISP is fully implemented for the major dialects of LISP and is available over the ARPANET. The technology transfer has been successful, and GLISP is now widely used. For more information, see HPP reports HPP-82-1[14], HPP-82-32[15], HPP-82-34[16], HPP-82-35[17], HPP-83-15[18].

3.2.2. AGE (Attempt to GEneralize): A Knowledge Engineering Tool

AGE is a skeletal system directed primarily at building blackboard application systems. Because the specifications of the blackboard global data structure, the knowledge sources, and the control mechanisms that manipulate the knowledge source invocations are interrelated, the normal linear editing facilities provided in AGE-1 turned out to be quite awkward. In addition, the many options available, especially in the control structure, confused the casual users. To alleviate some of these problems, we implemented a graphics-oriented editor using the bit-map displays available on Lips workstations (Xerox D-machines). The experiment was to see whether the ability to see the three major components of blackboard systems simultaneously, and in graphical form, would make the system easier to use. The graphic version, AGE-1.5, is, in fact, easier to use, and has been distributed to those people who are using Xerox machines.

The other problem, that of helping novice users with intelligent design aid, was not tackled. AGE-1 uses a relatively old (ten-years-old) technology, and it needs to be redesigned and reimplemented to make it worthwhile adding new capabilities. It is expected that industry AI tool makers will carry on in this vein. In the mean time, our efforts are directed at (1) conducting and writing a retrospective analysis of AGE-1 and its uses, and (2) experimenting with concurrent blackboard systems to speed up the execution of blackboard systems.

3.3. Elucidating the Nature of Heuristics

EURISKO uses a frame-based representation of starting assumptions and definitions in order to discover new knowledge in a domain through opportunistic search of the interesting combinations of the primitive concepts. It is a domain independent outgrowth of the AM (Automated Mathematician) work of Professor Douglas Lenat. EURISKO was successfully tested on domains as varied as war gaming and integrated circuit design architectures. It was co-funded by ONR Contract N00014-80-C-0609. Publications that relate to this work are HPP-81-22[19], HPP-82-25[20], and HPP-82-26[21].

More recent work on discovery has proceeded under the MOLGEN project, concentrating on the domain of regulatory molecular genetics and designing laboratory experiments. See HPP-82-22[22], KSL-85-6[23], KSL-85-7[24], KSL-85-36[25].

4. The KB-VLSI Project: The Palladio System

During the contract period the major focus of the KB-VLSI Project has been the development and refinement of the Palladio system. Palladio is an exploratory environment for experimenting with circuit and system design representations, design methodologies, and knowledge-based design aids. It differs from other prototype design environments in that it provides mechanisms for constructing, testing and incrementally modifying or augmenting design languages and design tools.

Palladio provides a testbed for investigating elements of circuit and system design that include specification, refinement, symbolic simulation, use of prior designs, and expert system design aids. It has facilities for conveniently defining models of circuit or system stucture and behavior. These models, called perspectives, are similar to design levels in that the designer can use them to interactively create and refine design specifications. Palladio provides an interactive graphics interface for displaying and editing structural perspectives of circuits or systems in a uniform, perspective-independent manner. A declarative, temporal logic behavioral language with an associated interactive behavior editor is used to specify designs from a behavioral perspective. Further, a generic, event-driven symbolic simulator can simulate and verify the behavior of a specified circuit or system from any behavioral perspectives and can perform hierarchical and mixed-perspective simulations.

During the project period we completed and demonstrated both intial and refined versions of the Palladio system, including example knowledge-based system design aids. The initial version of Palladio was implemented in Interlisp-D on a Xerox 1100 workstation. The refined version is implemented in Zetalisp on the Symbolics 3600 workstation. The latter version addresses, in particular, some of the efficiency problems of the initial implementation. For example, the Zetalisp version includes the automatic translation of behavioral specifications expressed as declarative logic assertions into Lisp procedures. Palladio uses the procedural form of behavior for efficiency in simulation of a circuit and the declarative form for reasoning about the circuit's behavior (e.g., for fault diagnosis and test generation).

The refined version of Palladio is being used as the basis for several research activities within the Knowledge Systems Laboratory at Stanford, at Schlumberger Palo Alto Research (SPAR), and at Lockheed's Palo Alto Research Center. For example, the system is being used by an advanced architecture project at Stanford to investigate concurrent hardware and software architectures for knowledge-based systems and by a group at Lockheed to investigate performance characteristics of a proposed dataflow architecture. Moreover, at least two industrial organizations are developing commercial design systems based on concepts demonstrated by the Palladio system.

The basic Palladio system is described in detail in HPP-83-31[26].

5. Intelligent Agents Project

5.1. Introduction

This section summarizes progress on basic and applied research in the study of interacting "intelligent agents" (IA's), each capable of acting autonomously in a precisely specified domain. The principal application domain was that of an intelligent interface to a computer operating system.

Work in this area split naturally into two broad subareas. The first involved construction of an individual intelligent agent, while the second required investigation of the issues involved in the interaction between a variety of such agents.

The construction of a single autonomous agent in a complex domain itself involved a variety of problems. First, planning research needed to be done to investigate the problems that will be encountered by a planner that needs both to observe its environment before making plans and to execute these conditionally, allowing for the possibility of failure along the way. Intelligent interaction with computer operating systems, including access to remote devices, requires that both of these problems be addressed.

Second, given these constraints, the domain is sufficiently complex that "blind" planning and inference are unlikely to manage the combinatorial difficulties of the problems encountered. Research was therefore required on the control of such inference, as the IA needs to be able to introspect and control its own activities.

Finally, an individual IA will need to present its results to the user. This led us to investigate the questions involved in intelligent presentation, as well as those in intelligent planning and controlled inference.

Interaction between IA's is a related but separate issue. Research in this area has had to address questions involving general problems of interaction between agents whose goals or world models may conflict; these questions had gone nearly untreated in the AI literature until the IA project addressed them.

5.2. Sensory and Conditional Planning

The Intelligent Agents domain is one in which many planning problems do not have guaranteed linear solutions. Instead, it is necessary to take steps to gather information about the state of the world and to then act upon the information gathered. This gives rise to plans which, instead of being a simple sequence of actions, contain conditional *IF-THEN-ELSE* constructs. In addition, such plans may also contain "sensory" actions, the purpose of which are to gather information rather than to change the state of the world. An approach to planning has been developed which allows the construction of conditional-sensory plans in a natural manner, including cases where the sensory actions themselves have prerequisites which must be achieved[27]. The formalism developed for planning has proven useful in other design problems [28, 29, 30], and has also given rise to research on efficiency gains made by exploiting existing design constraints[31, 32, 33]. Finally, since most planning and error recovery in the IA domain is fairly standardized, an "expert system" style planner was developed with extensive abilities to transfer files between numerous sites over three networks and to recover from a large number of commonly occurring errors[34, 35].

5.3. Control of Inference

As remarked in the introduction, the need to use more than "canned plans" in the IA domain led to a clear need for better and more general strategies for control of inference than had previously been available. The IA domain poses many problems in which there are large conjunctive queries to solve. If improperly ordered, these problems are often intractable, but use of knowledge about the number of answers for the individual conjuncts can lead to dramatic decreases in the number of possible answers which must be examined. A number of results were obtained on this problem[36, 37]. Another area of control research investigated the problem of deciding when a search could be stopped because all possible answers to the original query had already been found[36, 38, 39].

This work dovetails neatly with the more general work on control of inference which is also in progress at Stanford. An area of this more general control work which was investigated under the IA project involved studying the tradeoff between devoting resources in an attempt to find the best course of action in solving a problem and devoting those same resources to a direct attempt at a solution. A variety of situations were studied and a number of "break-even" points were discovered[40].

5.4. Intelligent Presentation

An IA's activity will frequently involve the presentation to a human user of informaion about the state of the IA's world. If, for example, a plan has failed unexpectedly, it is far better that the user be presented with a sketch of what has happened and where the world stands now rather than that the machine simply aborts the plan in some unknown state. Not surprisingly, however, the IA will have far more information about the state of the world than should be presented to the user, and the IA will therefore need to present only a suitable subset of the information available to it.

Flexibility in the nature of the information being presented requires flexibility in the method of presentation as well, since different sorts of facts should generally be presented using different representation methods. An investigation of these issues led us to explore a variety of criteria for automatic generation and evaluation of methods for machine presentation of information[41, 42, 43, 44].

5.5. Interacting Agents

The IA includes many "canonical agents" working together. However, it is not always possible or even desirable for the individual agents to blindly obey one another; to lay the foundation for an investigation of the details of multi-agent interaction, we began by defining notions of rationality for single agents. The first case investigated dealt with agents in a master/slave relation, so that each agent was able to fully specify the actions of other agents in order to carry out a task. A solution to the problem of ordering activities among agents was discovered[45].

The next problem considered dealt with the discovery of communication strategies between agents with identical goals but with incomplete or conflicting world models[46]. Following this, we were finally able to turn to the case of fully autonomous agents that would not necessarily cooperate but might instead pursue courses of action intended to achieve potentially conflicting goals. A hierarchy of rationality assumptions was developed, and we investigated the consequences of each agent's assuming that the behaviors of the others could be described by one or more of the definitions in the hierarchy. The formalism developed allowed for the modelling of restrictions on communication and the exchange of binding promises among agents. The work is described in[47, 48, 49, 50] Another area of research was on the tradeoff between communication costs and parallelism. It was discovered that the communication costs incurred in a joint effort may outweigh the advantages of parallel processing in some situations, and an efficient communication protocol called ESP was developed to address these problems. We also investigated a "Variable Supply Model," which covers a large spectrum of strategies in the tradeoff between parallelism and communication. A case study involving run-time allocation of deductions to multiple agents is presented in[51] and shows various empirical breakeven points in the trade between parallelism and reduced communication costs. Another case study is presented in[52] and investigates the static allocation of deductions to a large number of agents. This study attempts to make compile-time estimations of run-time communication costs between the agents and, based upon these estimations, to distribute the database among the various agents such that agents with a great need to communicate will be near to one another. In addition, the system can fold parts of the database together onto the same agent in an attempt to reduce communication costs at the expense of reduced parallelism. Relevant references are[53, 52, 51, 54, 55].

6. Blackboard Systems Research

1. We surveyed existing blackboard architectures (e.g., AGE, HEARSAY-III) and systems e.g., (HEARSAY-II, HASP, OPM, CRYSALIS) in an effort to define the fundamental components of the blackboard architecture, to characterize the conceptual and computational strengths of the architecture, and to identify important areas for further development of the architecture. We reported our conclusions in HPP-83-30, 1983[56] and[57].

2. We focused our subsequent research on elaborating the blackboard architecture in the key area of control: which action should an application system perform at each point in the problem-solving process? We formulated a set of behavioral goals for intelligent control, developed a uniform "blackboard control architecture" for achieving those goals, and demonstrated the architecture's applicability to the control strategies of HEARSAY-II, HASP, and OPM. We reported these results in HPP-83-38, 1984[58] and[59].

3. We implemented the blackboard control architecture as a domain-independent system-building environment called BB1. We augmented its control capabilities with related capabilities for explanation of problem-solving behavior and automatic acquisition of problem-solving strategies. We described BB1 and its early capabilities in HPP-84-16, 1984[60] and HPP-85-2, 1985[3].

4. We have continued to extend and refine BB1 to include a graphical user interface, knowledge-base management facilities, and capabilities for defining and interpreting task-specific problem-solving languages. We have distributed BB1 to approximately fifteen users outside of Stanford, including several DARPA contractors. We are in the process of implementing BB1 in Commonlisp (it currently is implemented in Interlisp), to make it accessible to a wider range of researchers. We are also preparing a BB1 users manual.

5. We are collaborating with other Stanford scientists to develop the PROTEAN system for protein-structure analysis within the BB1 architecture. This problem poses a number of technical problems for which the architecture is well-suited, for example: a large search space, the need to integrate qualitatively different sorts of knowledge, the need to reason about and explain problem-solving strategy, and the need to acquire problem-solving strategy automatically from domain experts. We have implemented a prototype system and are in the process of implementing a second-generation prototype. We have produced three articles[61],[2],[62].

6. In the course of our work on the PROTEAN system, we abstracted a domain-independent problem-solving framework that is applicable to a larger class of "arrangement problems:" arrange a set of objects in some context to satisfy a set of constraints. We have implemented this framework as ACCORD, a modular extension of the BB1 architecture. ACCORD is being used in the second-generation PROTEAN prototype mentioned above. We also plan to make it available to other researchers who are attempting to solve other arrangement problems. We are in the process of writing two technical reports, one that discusses the utility of problem-solving frameworks such as ACCORD and one that discusses the ACCORD framework per se.

7. We are exploring other application areas for BB1, including engineering design and real-time process control.

7. Advanced Knowledge-based System Architectures

In 1983, the HPP turned its attention to a research area it had first studied in the Contract Nets work of the mid-1970s: seeking big increases in the speed of symbolic processing and problem solving by exploiting multiprocessor architectures with distributed memory and control. The Advanced Architectures project is now a part of the national DARPA Strategic Computing Program. The project is long-range in nature, and has two related goals:

- To realize a new generation of software system architectures using parallelism to achieve high-speed computation in artificial intelligence applications.

- To specify multiprocessor hardware system architectures that support those parallel computations.

Task Description

The basic problem we are addressing is to increase the speed of execution of expert systems through the use of parallel computations on a multiprocessor computer system. Part of the effectiveness of expert systems, particularly for real-time applications such as continuous signal data understanding, lies in the speed of execution, or throughput rate. However, for many significant applications of this type, projected performance limits of uniprocessors fall short of the speed required by as much as several orders of magnitude. Multiprocessor parallel computing must be used to attain the necessary levels of performance.

The anticipated computational requirements for the next decade cannot be realized by just using parallelism at only one particular level of computation (for example, parallel left-hand-side rule matching in rule-based systems). Rather, parallelism must be supported at all levels of the computational hierarchy. At each level, we seek to use parallelism to increase speed by factors of from two to ten. By careful organization of the computational hierarchy and the hardware system architecture, these small gains must be made to reinforce each other multiplicatively.

Approach

At present, little is known about the qualitative aspects of programming expert systems within a multiprocessor setting or about the potential quantitative gains resulting from multiprocessor-based expert systems. To understand the effectiveness of parallel implementations of expert systems, one must study both the programming problems and the performance issues at all levels of the computational hierarchy. Our research emphasis is therefore on overall software and hardware system architectures for the parallel execution of expert systems.

In keeping with the empirical approach that has characterized HPP research, we focussed on a specific class of applications, taking a vertical slice through the space of design alternatives at each level of the hierarchy:

- The application classes.
- The problem-solving framework and knowledge representations.
- The programming languages.
- The hardware system architectures.

The Application Classes. We have chosen to study the class of applications that involves signal understanding, information fusion, and situation assessment. Specific applications of this class for which expert systems have been developed include vision and speech understanding systems and programs for understanding passive and active sonar and radar data. Such applications have several characteristics that lend themselves to parallelism, including diverse, independent sources of knowledge and solutions that are formed incrementally over time. In addition, they often admit the processing of information in a pipelined manner, that is, the use of concurrent suboperations for a stream of data.

In the first phase of the research (through September 1985) we worked primarily with an existing expert system for interpretation of passive radar signals (ELINT). In later phases we intend to develop new and more complex applications, within the selected application class, in order to validate and tune our system architectures.

The Problem-Solving Framework. We have chosen to use blackboard-based systems as our problem-solving framework. Blackboard frameworks have intrinsic parallelism in their multiple, independent knowledge sources and distributed control. Since they were developed in previous work on applications of the type we are exploring, they have particularly appropriate characteristics. For example, they incorporate hierarchical organization of the solution state as well as admitting diverse forms of knowledge, the use of an evolving solution, and both data- and model-driven reasoning.

The blackboard is also well suited to exploit concurrency at various levels of processor and process "granularity" that is, the size and number of processors and the size of the tasks assigned to each processor. At a coarse level of granularity, a blackboard system can be broken down into multiple blackboards and sub-blackboards assigned to distinct processors. 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 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 separate processors. At finer levels, each knowledge source could be associated with one or more processors for parallel execution, and the individual entities on the blackboard could be treated as active objects and assigned to distinct processors. Such an assignment of blackboard objects to processors would permit, for example, parallel blackboard search.

In organizing the application for parallel execution within a blackboard framework, a number of issues need to be investigated a global versus a distributed blackboard, process granularity, replication of knowledge sources, and trade-offs between levels of control and parallelism. To understand the speedup potential of such organizations, we are studying the relationships between knowledge sources and the areas of the blackboard that they touch, the penalties involved in replication of portions of the blackboard, and the inherent sequential nature (or lack of it) of the reasoning processes used. This plan of attack was formulated during this reporting period, with the intention of executing it during the follow-on research sponsored by DARPA within the SCP.

The Programming Language. We have chosen Lisp-based languages for programming because Lisp contains a very rich set of constructs for symbolic processing and can be easily used to implement other programming styles, such as object-oriented or declarative logic. Moreover, Lisp is a mature language and its implementation trade-offs are well understood.

The parallel programming language issues that we are investigating include the trade-off between shared and distributed address spaces; trade-offs between communication costs, process establishment costs, and concurrency; and techniques for process and data resource allocation and reclamation.

The Hardware System Architecture. Rather than building hardware prematurely, we are using simulations and other modeling techniques to develop our understanding of the types of multiprocessor hardware systems needed to support the parallel execution of expert systems. The HELIOS simulation technology developed in the KBVLSI project was extended to handle the need to simulate a general multiprocessor architecture. The hardware system simulator includes dynamic, graphics-based tools for evaluating performance. An important part of our research effort is the development both of methods for evaluating multiprocessor performance and of the supporting tools for this work.

Our initial focus is on grid-based array architectures, where each node in the array consists of a processor for managing communications and process scheduling, a processor for performing Lisp evaluations, and local memory. Interprocessor communication in the grid is via message-passing protocols. Our simulation model of this architecture is defined with specifiable parameters to permit the investigation of, for example, different interconnection topologies for the grid and various processor and communication characteristics.

The hardware system architecture issues that we are investigating include the sizes and types of processors and memories, the interprocessor communication topologies and mechanisms, and the organization of memories with respect to processors.

Current Status (1985)

The Advanced Architectures project was still in its beginning stages at the conclusion of this contract period. We have implemented a simulation model of the grid-based array hardware system and an initial set of performance evaluation tools for our simulator. We have developed and implemented a high-level parallel object-oriented extension of QLAMBDA, which executes on our simulated multiprocessor hardware system. We have begun conducting several experiments investigating the potential concurrency in blackboard-based expert systems in signal understanding applications using various levels of process and data granularity. In addition, we have defined and implemented a parallel-processing version of our blackboard framework software development system, called CAGE (Concurrent AGE).

We initiated a series of "vertical slice" experiments, which consist of simulated parallel executions of a passive radar signal understanding expert system. Although quantitative results were not available by the conclusion of this reporting period, initial results indicate that there is significant exploitable parallelism in expert systems of this type.

8. KSL Computing Resources

The research of the Knowledge Systems Laboratory is heavily dependent on computing resources tuned to support the dialects of Lisp that are used by our research staff (InterLisp, MACLisp, ZetaLisp, FranzLisp, and CommonLisp). These computing systems must, in addition to supplying adequate computing capacity, support our memory and graphics needs and provide program development and debugging environments that facilitate the construction and testing of the large AI systems under study in the KSL. Since our first experiments with them in the early 80's, Lisp workstations are rapidly becoming the primary source of computing power for the KSL. These machines are the highest-performance Lisp engines available and have very large address spaces, flexible graphics interfaces for users, state-of-the-art program development and debugging tools, and a modularity that will be the vehicle for disseminating AI systems into many user environments.

Over the three years of this contract, we have used funds allocated for equipment purchases primarily for new Lisp machines. The funds from this contract were used in conjunction with equipment purchase funds from other sources to negotiate vendor gifts to the KSL to maximize the equipment purchased per dollar. Our purchases include:

- 7 Texas Instruments Explorers (Model 2249426-0025) with 8 MBytes of memory, 140 MByte disks, and Ethernet interfaces.
- 2 Symbolics Lisp Machines (1 model 3640 and 1 model 3600) with 8 MBytes of memory, 169 MByte disks, and Ethernet interfaces.
- 7 Xerox Lisp Machines (model 1186-102) with 3.7 MBytes of memory, 40 MByte disks, and Ethernet interfaces.
- 10 Zenith terminals (model Z-29)

In the transition from mainframes to workstations, much system development must be done. This includes integrating the workstations into a network environment with shared resources such as printers and file servers and replacing the full range of tools such as text processing, electronic mail, file manipulation, budget preparation and control, drawing and so on that now keep workstation users tethered to expensive and overloaded mainframe systems. But it also includes extensions so that users can interact more effectively with their computing environment through more intelligent customized interface agents, can access graphics-based systems and tools remotely (from home and more distant locations), can use effective and transparent network services (file storage, printing, gateway routing, etc.), and can take advantage of the networked concurrent architecture these workstations represent to reaggregate the large combined computing capacity inherent in large groups of distributed workstations. These system software developments are now actively underway.

References

1. Bennett, J., "Roget: A knowledge-based system for acquiring the conceptual structure of an expert system," Technical Report HPP-83-24, Knowledge Systems Laboratory, Stanford University, October 1983.

2. Buchanan, B., Altman, A., Brinkley, J., Cornelius, C., Duncan, B., Hayes-Roth, B., Hewett M., Lichtarge, O., and Jardetzky, O., "A new method for deriving solution structures of proteins from NMR data," Technical Report KSL-85-41, Knowledge Systems Laboratory, Stanford University , October 1985, To appear in Proceedings of the National Academy of Science

3. Hayes-Roth, B. and Hewett, M., "Learning control heuristics in BB1," Technical Report KSL-85-2, Knowledge Systems Laboratory, Stanford University, January 1985, Submitted to the IJCAI-85. Also STAN-CS-85-1036

4. Ginsberg, M.L., "Analyzing incomplete information," Technical Report HPP-84-17, Knowledge Systems Laboratory, Stanford University, 1984.

5. Dietterich, T.G., "Constraints propagation techniques for theory-driven data interpretation," Technical Report HPP-84-46, Knowledge Systems Laboratory, Stanford University, December 1984, PhD Thesis, to be published as a book by Kluwer

6. Fu, Li-Min, "Learning object-level and meta-level knowledge in expert systems," Technical Report KSL-85-44, Knowledge Systems Laboratory, Stanford University, November 1985, PhD Thesis

7. Griener and Genesereth, "The role of abstractions in understanding analogy," Technical Report HPP-84-8, Knowledge Systems Laboratory, Stanford University, April 1984.

8. Wilkins, D.C., Buchanan, B.G. and Clancey, W.J., "Inferring an expert's reasoning by watching," Technical Report HPP-84-29, Knowledge Systems Laboratory, Stanford University, 1984, To appear in Proceedings of the 1984 Conference on Intelligence Systems and Machines

9. Wilkins, D.C., Clancey, W.J. and Buchanan, B.G., "An overview of the Odysseus Learning Apprentice," Technical Report KSL-85-26, Knowledge Systems Laboratory, Stanford University, August 1985, To appear in Machine Learning: A Guide to Current Research, Academic Press, 1986, in press

10. Rosenbloom, P.S., Laird, J.E., McDermott, J., Newell, A. and Orciuch, E., "R1-Soar: An experiment in knowledge-intensive programming in a problem-solving architecture," Technical Report HPP-84-39, Knowledge Systems Laboratory, Stanford University, October 1984, Appeared in the Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems

11. Laird, J. Rosenbloom, P. and Newell, A., "Chunking in Soar: The anatomy of a general learning mechanism," Technical Report KSL-85-34, Knowledge Systems Laboratory, Stanford University, September 1985, Appears in Machine Learning, Vol. 1, No. 1. Also appears as Xerox PARC ISL-13 and Carnegie-Mellon University-CS-85-154

12. Haggerty, J., "REFEREE and RULECRITIC: Two prototypes for assessing the quality of a medical paper," Technical Report HPP-84-49, Knowledge Systems Laboratory, Stanford University, 1984, Master's Thesis

13. Buchanan, B.G., "Some approaches to knowledge acquistion," Technical Report KSL-85-38, Knowledge Systems Laboratory, Stanford University, October 1985, Appears in Proceedings of the Third International Workshop on Machine Learning

14. Novak, G.S. Jr., "GLISP user's manual," Technical Report HPP-82-1, Knowledge Systems Laboratory, Stanford University, January 1982.

15. Novak, G., "The GEV display inspector/editor," Technical Report HPP-82-32, Knowledge Systems Laboratory, Stanford University, November 1982.

16. Novak, G., "Data abstraction in GLISP," Technical Report HPP-82-34, Knowledge Systems Laboratory, Stanford University, December 1982, Also appears in Proceedings of SIGPLAN '83, Symposium on Programming, 1983.

17. Novak, G., "GLISP: A high-level language for A.I. programming," Technical Report HPP-82-35, Knowledge Systems Laboratory, Stanford University, August 1982, Also appears in Proceedings of the Second National Conference on Artificial Intelligence

18. Novak, G.S., "Knowledge-based programming using abstract data types," Technical Report HPP-83-15, Knowledge Systems Laboratory, Stanford University, March 1983, To appear in the Proceedings of the AAAI, pp. 288-291, August 1983

19. Lenat, D., "The nature of heuristics," Technical Report HPP-81-22, Knowledge Systems Laboratory, Stanford University, 1981, Also appears in Artificial Intelligence 19, pp. 189-249, 1982

20. Lenat, D., "Nature of heuristic II," Technical Report HPP-82-25, Knowledge Systems Laboratory, Stanford University, March 1982, Also appears in Artificial Intelligence Journal, Vol. 21, 1:2

21. Lenat, D., "EURISKO: A program that learns new heuristics and domain concepts," Technical Report HPP-82-26, Knowledge Systems Laboratory, Stanford University, March 1982, Also appears in Artificial Intelligence Journal, Vol. 21, 1:2

22. Iwasaki, I., "SPEX: Skeletal planner of experiments," Technical Report HPP-82-22, Knowledge Systems Laboratory, Stanford University, September 1982.

23. Friedland, P.E. and Iwasaki, Y., "The concept and implementation of skeletal plans," Technical Report KSL-85-6, Knowledge Systems Laboratory, Stanford University, 1985, Also appears in the Journal of Automated Reasoning, 1985

24. Bach, R., Iwasaki, Y., and Friedland, P., "Intelligent computational assistance for experiment design," Technical Report KSL-85-7, Knowledge Systems Laboratory, Stanford University, 1985, Also appears in Nucleic Acids Research, 1985

25. Karp, P., "Thesis proposal: Qualitative simulation and discovery in molecular biology," Technical Report KSL-85-36, Knowledge Systems Laboratory, Stanford University, September 1985, Working paper

26. Brown, H., Tong, C. and Foyster, G., "Palladio: An exploratory environment for IC design," Technical Report HPP-83-31, Knowledge Systems Laboratory, Stanford University, June 1983, Also appears in IEEE Computer-83, pp. 41-56, (0018-9162/83/1200-0041)

27. Finger, J. J., "Sensory Planning," Tech. report HPP-82-12, Stanford University, April

1982.

28. Genesereth, Michael R., ed., "The MRS Casebook," Tech. report HPP-83-26, Stanford University, May 1983.

29. Genesereth, Michael, "Partial Programs," Heuristic Programming Project Memo HPP-84-2, Stanford University, November 1984.

30. Genesereth, Michael R., "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence*, Vol. 24, 1984, pp. 411-436.

31. Finger, J. J. and Michael R. Genesereth, "RESIDUE - A Deductive Approach to Design," Tech. report HPP-83-46 (working paper), Stanford University, December 1983.

32. Finger, J. J. and Michael R. Genesereth, "RESIDUE - A Deductive Approach to Design Synthesis," Tech. report HPP-85-1, Stanford University, January 1985.

33. Finger, Joseph Jeffrey, *Exploiting Design Constraints*, PhD dissertation, Stanford University, 1986, (in preparation)

34. Dolbec, Michael, "A Planner for the Intelligent Agent", Master's Practicum

35. Cronin, Jonathan, "The Intelligent Agent: An Expert System", Master's Practicum

36. Smith, D. E., *Controlling Inference*, PhD dissertation, Stanford University, August 1985.

37. Smith, D. E. and Genesereth, M. R., "Ordering Conjunctive Queries," *Artificial Intelligence*, Vol. 25, 1985.

38. Smith, D. E., "Finding All of the Solutions to a Problem," *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, August 1983.

39. Smith, D. E. and Genesereth, M. R., "Controlling Recursive Inference," Heuristic Programming Project Memo HPP-84-6, Stanford University, 1984.

40. Rosenschein, Jeffrey S. and Singh, Vineet, "The Utility of Meta-level Effort," Report No. HPP-83-20, Heuristic Programming Project, Computer Science Department, Stanford University, March 1983.

41. Mackinlay, Jock, "Intelligent Presentation: The Generation Problem for User Interfaces," Heuristic Programming Project Memo HPP-83-34, Stanford University, March 1983.

42. Mackinlay, Jock and Michael R. Genesereth, "Expressiveness of Languages," *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August 1984.

43. Mackinlay, Jock D., *Automatic Design of Graphical Presentations*, PhD dissertation, Stanford University, 1986, (in preparation)

44. Mackinlay, Jock D. and Genesereth, Michael R., "Expressiveness and Language Choice," *Data & Knowledge Engineering*, Vol. 1, 1985, pp. 17-29.

45. Rosenschein, Jeffrey S., "Synchronization of Multi-Agent Plans," *Proceedings of The National Conference on Artificial Intelligence*, The American Association for Artificial Intelligence, Pittsburgh, Pennsylvania, August 1982, pp. 115-119.

46. Rosenschein, Jeffrey S. and Genesereth, Michael R., "Communication and Cooperation,"

HPP Report 84-5, Heuristic Programming Project, Computer Science Department, Stanford University, March 1984.

47. Michael R. Genesereth and Matthew L. Ginsberg and Jeffrey S. Rosenschein, "Cooperation without Communication," HPP Report 84-36, Heuristic Programming Project, Computer Science Department, Stanford University, September 1984.

48. Michael R. Genesereth and Matthew L. Ginsberg and Jeffrey S. Rosenschein, "Solving the Prisoner's Dilemma," HPP Report 84-41, Heuristic Programming Project, Computer Science Department, Stanford University, November 1984.

49. Rosenschein, Jeffrey S. and Michael R. Genesereth, "Deals Among Rational Agents," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* The International Joint Conferences on Artificial Intelligence, Los Angeles, California, August 1985, pp. 91-99, Also published as HPP Report 84-44, Heuristic Programming Project, Computer Science Department, Stanford University, December 1984

50. Rosenschein, Jeffrey Solomon, *Rational Interaction: Cooperation Among Intelligent Agents,* PhD dissertation, Stanford University, October 1985.

51. Singh, Vineet and Michael R. Genesereth, "A Variable Supply Model for Distributing Deductions," *Proceedings of IJCAI-85,* Morgan Kaufmann Publishers Inc., August 1985, Submitted to Journal of Parallel and Distributed Computing

52. Singh, Vineet and Michael R. Genesereth, "PM: A Parallel Execution Model for Backward-Chaining Deductions," KSL Report KSL-85-18, Stanford University, May 1985, Submitted to Future Computing Systems

53. Singh, Vineet, *Distributing Deductions to Multiple Processors,* PhD dissertation, Stanford University, June 1986.

54. Vineet Singh and Michael R. Genesereth, "A Variable Supply Model for Distributing Deductions," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* The International Joint Conferences on Artificial Intelligence, Los Angeles, California, August 1985, pp. 39-45.

55. Singh, Vineet and Genesereth, Michael R., "A Variable Supply Model for Distributing Deductions," Tech. report HPP-84-14, Heuristic Programming Project, Computer Science Department, Stanford University, May 1984.

56. Hayes-Roth, B., "The blackboard architecture: A general framework for problem solving," Technical Report HPP-83-30, Knowledge Systems Laboratory, Stanford University, May 1983.

57. Hayes-Roth, B., *Encyclopedia of Artificial Intelligence, D. Eckroth (ed.),* John Wiley & Sons, New York, 1985, In press

58. Hayes-Roth, B., "The blackboard model of control," Technical Report HPP-83-38, Knowledge Systems Laboratory, Stanford University, June 1983.

59. Hayes-Roth, B., "A blackboard architecture for control," *Artificial Intelligence,* Vol. 26, 1985, pp. 251-321.

60. Hayes-Roth, B., "BB1: An architecture for blackboard systems that control, explain, and learn about their own behavior," Technical Report HPP-84-16, Knowledge Systems Laboratory, Stanford University, December 1984.

61. Hayes-Roth, B., Buchanan, B.G., Lichtarge, O., Altman, R., Hewett, M., Brinkley, J., Cornelius, C., Duncan, B. and Jardetzky, O., *Elucidating Protein Structure From Constraints in PROTEAN, R. Engelmore and A. Morgan (eds.),* Addison-Wesley, Berkshire, 1986, In press

62. Jardetzky, O., Lane, A., Lefevre, J., Lichtarge, O., Hayes-Roth, B. and Buchanan, B., "Determination of macromolecular structure and dynamics by NMR," *Proceedings of the NATO Advanced Study Institute "NMR in the Life Sciences",* Plenum Publishing Corporation, 1985, In press

KSL Report List

A complete list of KSL reports can be requested from:

Knowledge Systems Laboratory
701 Welch Road, Bldg. C
Palo Alto, CA 94304