Artificial Intelligence Research in the HEURISTIC PROGRAMMING PROJECT

Stanford University

THE HEURISTIC PROGRAMMING PROJECT (HPP) is a research laboratory of about 70 people—faculty, staff, and graduate students—within the Computer Science Department at Stanford University. It is under the overall direction of Professors Edward A. Feigenbaum and Bruce G. Buchanan as senior principal investigators and Thomas C. Rindfleisch as project director. For 18 years, beginning with DENDRAL in 1965, HPP has concentrated on research that explores the mechanization of symbolic reasoning and problem solving processes that are based on extensive domain-specific knowledge. Our approach has been to work on significant real-world applications, carefully chosen from domains such as science, medicine, engineering, and education, that expose key, underlying AI research issues.

For HPP, AI is largely an empirical science Research questions are explored by designing and building programs that incorporate plausible answers. These hypotheses are tested by experimenting with perturbations of the systems and extrapolations into new problem areas. The test of success in this endeavor is whether the next generation of system builders finds the questions relevant and the answers applicable for understanding and building more complex reasoning programs.

The basic research issues at the core of HPP's interdisciplinary approach center on the computer representation and use of large amounts of domain-specific factual and judgmental knowledge. These questions have long guided HPP work and are now of central importance to other AI research as well:

1 Knowledge representation: How can the knowledge necessary for complex problem solving be represented for its most effective use in automatic inference processes? Often, the knowledge obtained from experts is heuristic knowledge, gained from many years of experience How can this knowledge, with its inherent vagueness and uncertainty, be represented and applied?

2. Knowledge acquisition: How is knowledge acquired most efficiently—from human experts, from observed data, from experience, and from discovery? How can a program discover inconsistencies and incompleteness in its knowledge base? How can the knowledge base be augmented with only appropriate perturbations to the established knowledge base?

3 Knowledge utilization: By what inference methods can many sources of knowledge of diverse types be made to contribute jointly and efficiently toward solutions? How can knowledge be used intelligently, especially in systems with large knowledge bases, so that it is applied in an appropriate manner at the appropriate time?

4 Explanation and tutoring: How can the knowledge base and the line of reasoning used in solving a particular problem be explained to users? What constitutes a sufficient or an acceptable explanation

for different classes of users? How can problem solving systems be combined with pedagogical and user knowledge to implement intelligent tutoring systems?

5. System tools and architectures: What kinds of software tools and system architectures can be constructed to make it easier to implement expert programs with increasing complexity and high performance? What kinds of systems can serve as vehicles for the cumulation of knowledge of the field for the researchers?

Numerous high-performance, knowledge-based programs have resulted from this work in such diverse fields as analytical chemistry, infectious disease diagnosis, cancer chemotherapy management, VLSI design, machine fault diagnosis, and molecular biology. Some of these programs rival human experts in solving problems in particular domains and some are even being adapted for commercial use. Other results include generalized software tools for representing and utilizing knowledge (e.g., EMYCIN, Units, AGE, MRS, and GLISP) as well as comprehensive publications like the three-volume *Handbook of Artificial Intelligence* (Barr and Feigenbaum, 1981, 1982; Cohen and Feigenbaum, 1982) and the forthcoming book on the MYCIN experiments (Buchanan and Shortliffe, 1983).

Thus, the work of HPP can be characterized in three ways—by the expert systems it writes, by the system architectures and tools it creates, and by the basic research issues that motivate the work. The following paragraphs give brief overviews of major HPP projects currently in progress with pointers to more detailed published descriptions Indicated personnel include directly related staff and PhD candidates. In addition to the major projects, there is ongoing PhD thesis research by Daniel Berlin, Paul Cohen, Greg Cooper, John Kunz, and Robert London

References

Barr, A , and Feigenbaum, E A (1981, 1982) *The Handbook of Artificial Intelligence* (Vols 1 and 2). Los Altos, CA: Wm. Kaufmann

Cohen, P R., and Feigenbaum, E A (1982) *The Handbook of Artificial Intelligence* (Vol. 3). Los Altos, CA: Wm. Kaufmann.

Buchanan, B G , and Shortliffe, E. H (1983) *Expert systems research: The MYCIN experiments of the Stanford Heuristic Programming Project* Reading, MA: Addison-Wesley (forthcoming)

The EURISKO Project

EURISKO is a project that investigates machine learning, specifically the automatic discovery of new domain objects, operators, and heuristics Several hundred heuristics provide the judgmental knowledge to guide EURISKO in proposing plausible new terms, gathering data about them, noticing regularities, formulating hypotheses, designing experiments to test them, and shortening the statement of the conjectures by defining new terms—which starts the cycle over again. In addition, EURISKO occasionally uses its heuristics to "compile its hindsight" into new heuristics—rules that, if only it had had them sooner, would have enabled it to avoid many of the blind alleys it wasted time on and to focus more quickly on the concepts that turned out to be important. Even more rarely, EURISKO proposes extensions to its framelike representation, such as new kinds of slots to be defined, that will facilitate the learning process.

To date, we have applied EURISKO to elementary mathematics, naval ship design (where it has reigned undefeated in the Traveller TCS tournament), VLSI design (where it has come up with a few novel and potentially useful three-dimensional devices), programming (where it has uncovered a couple of LISP bugs), oil-spill cleanup, and a few other domains. Future work will include (a) examining the theoretical nature of heuristics - how and why they work, how they organize, and how they might be better represented; (b) applying EURISKO to more domains; (c) having EURISKO discover analogies both within and across the domains it applies to; (d) adding a very broad knowledge base to facilitate the search for and use of analogies; (e) modeling the user, both as an individual and as a member of various groups, to facilitate dialogues with him or her, and to learn new heuristics about various groups of people; and (f) getting EURISKO to produce more sophisticated changes of representation than the simple slot-definitions it currently does.

Personnel. Douglas Lenat (contact)

References

Lenat, D (1982) The nature of heuristics. AI *Journal* 19:2

Lenat, D. (1983) The nature of heuristics II AI *Journal* 20:2

Lenat, D (1983) The nature of heuristics III AI *Journal* 21:1-2

The Advanced Architectures Project

The Advanced Architectures project is a new, long-range research project to define and realize a new generation of computer architectures that provide high-speed computation for artificial intelligence applications. Specifically, we aim to design a system to facilitate building and running expert systems, involving the understanding of signals and situations.

It is our assumption that the projected computational requirements for the next decade cannot be realized by just speeding up the sequential machine or by the utilization of parallelism in any one particular level in a computational hierarchy. We propose to begin by conducting an empirical evaluation of the parallelism yielded in a class of applications, problem-solving frameworks, knowledge storage and access methods, reasoning mechanisms, implementation languages, and primitive data structures. The resulting measurements should provide the basis for work in designing sound, integrated hardware/software systems

One distinguishing concept for the research proposed here is that the eventual architecture is conceived in an environment that spans levels in a computational hierarchy from circuits through AI application systems. Initially, there will be much greater emphasis on the latter than on the former. That is, the research strategy is a top-down push from application to problem-solving frameworks and implementation languages and on to primitive data structures and operators that, in turn, can guide the hardware-level design. We feel that it is better to develop an architecture targeted at a specific, interesting class of applications than to develop a general-purpose logic engine. We are looking for a first-order complete solution to future knowledge-system problems.

Personnel. Edward A. Feigenbaum (contact), Bruce Delagi (DEC), Penny Nii, Richard Weyhrauch, Tom Rindfleisch

The Knowledge Acquisition Project

One of the critical research problems of AI is finding efficient means of building new knowledge bases. The process of transferring knowledge into a program can take many forms, from manual editing of data structures to automatic induction of rules from examples Currently, the most efficient method for giving a program world-class knowledge is knowledge engineering. But because of the time and manpower required for knowledge engineering to be successful, we are looking for tools to facilitate the process and methods that complement it.

There are many dimensions to the general problem, ranging from theory formation and conceptualization to debugging and editing. We are currently working on the following approaches.

1 Induction: Learning from positive and negative examples is an important source of new knowledge. Large databases contain valuable records of past experience that we would like to exploit in building a knowledge base. Induction is itself a knowledge-based task, so we are trying to understand the relationship between a syntactic generator of possible rules and knowledge-based (semantic) constraints on the generator We are beginning a project of this sort in the domain of medicine. Formulating student models in GUIDON and LMS is also largely an inductive task

2. Experimentation: When a learning program can actively experiment with the system it is trying to understand, it should be able to learn more effectively than when training instances are presented passively. We are building a program that experiments with file commands in the UNIX operating system in order to model the structure and behavior of that part of the system.

3. Analogy: Once a knowledge base is built, it should serve as a model for other knowledge bases in similar task domains Finding plausible analogies and using them in appropriate ways are the main topics of exploration. We are using knowledge bases built in MRS as analogues for new ones.

4. Watching: Another important method of gathering new information is to learn by watching an expert solve problems. This requires some familiarity with the vocabulary and structure of the domain and requires asking some questions of the expert. As the expert solves a problem, the learning program builds a model of the rules the expert seems to use, asking selected questions for clarification This work is being done in the context of the NEOMYCIN program, exploiting the user-modeling programs constructed for GUIDON

5. Dialogue. In many of the projects within HPP, we are improving the abilities of programs to communicate with both knowledge engineers and experts through graphics, smart editors, debugging tools, and knowledge base analysis tools The ROGET program is an experiment in putting knowledge engineering expertise into an expert system to aid in structuring knowledge initially for a new EMYCIN system.

Previous accomplishments in this area include META-DENDRAL, TEIRESIAS, and AM Other relevant research is being conducted in the EURISKO and RX projects, with still more projected in the context of ONCOCIN, MOLGEN, and DART.

In addition to the approaches mentioned above, one new direction will be synthesizing results from several different journal articles on the same topic into a coherent judgment about the state of knowledge on that topic. We will be working with highly stylized articles in medical journals reporting statistical results of clinical trials.

Personnel. Bruce Buchanan (contact), Avron Barr, Thomas Dietterich, Russell Greiner, Li-Min Fu

References

Davis, R , and Lenat, D. (1980) *Knowledge-based systems in artificial intelligence* New York: McGraw-Hill

Dietterich, T. G., and Buchanan, B G. (1983) The role of experimentation in theory formation Memo HPP-83-22, Heuristic Programming Project, Computer Science Department, Stanford University.

Lindsay, R., Buchanan, B. G., Feigenbaum, E A , and Lederberg, J (1980) *Applications of artificial intelligence for organic chemistry The DENDRAL project.* New York: McGraw-Hill

The ONCOCIN Project

ONCOCIN is an oncology protocol management system that has been in use since May 1981 by oncology faculty and fellows in the cancer chemotherapy clinic at the Stanford University Medical Center. Its research goals are directed both toward the basic science of artificial intelligence and toward the development of a clinically useful consultation tool. The AI research aims include:

1. The implementation and evaluation of recently developed techniques for making computer technology more natural and acceptable to physicians In particular, we are using professional workstations with advanced graphics capabilities to determine their potential utility as a vehicle for making high-performance consultation systems available to physicians cost-effectively
2. The extension of the methods of rule-based consultation systems so that they can interact with a large database of historical clinical information.
3. The continuation of basic research into areas such as mechanisms for handling time relationships, inexact reasoning, knowledge acquisition, explanation, and assessment of knowledge-base completeness and consistency.

Although it initially drew on the programs and capabilities developed for the EMYCIN system-building project, ONCOCIN is now largely a new program in which production rules are only one of several types of knowledge representation used. The system architecture is complex in that the program depends upon several simultaneous parallel processes that are highly interdependent The approach has been formalized in a scheme known as the Interviewer/Reasoner model.

The program has been customized to the specialized needs of an active clinical environment. ONCOCIN is an ongoing component of the oncology clinic's lymphoma treatment program. The actual use of ONCOCIN by physicians has led to several revisions in the underlying methodology and in the mode of interaction. It is used on a high-speed terminal with a specially designed keyboard and display interface As with other expert systems developed at HPP, ONCOCIN can give the reasons for its therapy recommendations. It is also able to use information regarding previous therapy in assessing how to treat the patient on subsequent visits. Recently, work has begun to transfer ONCOCIN to LISP machines; graphics capabilities will significantly improve the nature of the interaction between the physician and the program

Personnel. Ted Shortliffe, Charlotte Jacobs, Larry Fagan (contact), Miriam Bischoff, Bob Carlson, Christopher Lane, Curt Langlotz

References

Gerring, P. E , Shortliffe, E H , and van Melle, W. (1982) The Interviewer/Reasoner model: An approach to improving system responsiveness in interactive AI systems *AI Magazine* 3(4):24-27.

Langlotz, C. P., and Shortliffe, E. H. (1983) Adapting a consultation system to critique user plans. *International Journal of Man-Machine Studies.*, in press

Shortliffe, E H , Scott, A C , Bischoff, M. B , Campbell, A B., van Melle, W , and Jacobs, C D (1981) ONCOCIN: An expert system for oncology protocol management. *In proceedings 7th IJCAI, Vancouver, B.C.*, 876-881.

Suwa, M., Scott, A. C., and Shortliffe, E. H. (1982) An approach to verifying completeness and consistency in a rule-based expert system. *AI Magazine* 3(4):16-21.

The DART Project

The goal of the DART project is the automated diagnosis of equipment failures. The project is supported by IBM and began in late 1980. The primary result of the research to date has been a program embodying a new algorithm for diagnosing computer hardware faults. The program (called DART for Diagnostic Assistance Reference Tool) takes as input a set of observed symptoms, suggests tests and accepts the results, and ultimately pinpoints the fault responsible for the symptoms.

The distinguishing characteristic of DART is its use of design knowledge about the malfunctioning device Programs like CASNET, INTERNIST, and MYCIN all utilize "rules" or "facts" that associate symptoms with possible diseases. The DART program contains no information about how computers fail. Instead, it works directly from information about intended structure (a machine's parts and their interconnections) and expected behavior (equations, rules, or procedures that relate inputs and outputs). The advantage of this approach is that it greatly simplifies the task of building diagnostic systems for new devices. If a designer uses a modern computer-aided design system, then, when he is done, his design will be on-line. The structural and behavioral information in this model can then be passed as data to the DART program to diagnose its faults.

The idea of using design information in automated diagnosis is hardly a new one. Over the years, a number of test-generation algorithms have been proposed, the best known of which is the d-algorithm. The primary disadvantage of these algorithms is their specialization. The d-algorithm, for example, is based on Boolean algebra (or, more precisely, Roth's "d-calculus"), and so it is applicable only to devices whose behavior can be characterized in terms of bits. In contrast, DART uses a device-independent deductive procedure in diagnosing faults. The result is a program that can be used in diagnosing arbitrary devices or devices described in arbitrary ways. Interestingly, this generality also leads to efficiency. For example, it can be used to diagnose devices at multiple levels of abstraction and can thereby exploit hierarchy in a device's design to reduce the cost of diagnosis

Current subprojects include applications of the program to a wider variety of devices (including analog circuits and non-electronic devices), extensions of the program that allow it to improve its efficiency as a result of experience, the construction of a design-entry workstation to assist designers in getting their designs on line (including interactive graphics, simulation, and a program that suggests design modifications to enhance diagnosability), and a consultant to help field engineers carry out diagnostic and repair tasks.

Personnel: Michael Genesereth (contact), Milt Grinberg, Narinder Singh, Chuck Paulson

References

Genesereth, M R (1981) The use of design models in automated diagnosis. Memo HPP-81-20, Heuristic Programming Project, Computer Science Department, Stanford University.

Genesereth, M. R (1982) Diagnosis using hierarchical design models it In Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, 278-283

The NEOMYCIN/GUIDON2 Project

NEOMYCIN is a medical consultation system derived from MYCIN. The goal of the project is to develop a knowledge representation and explanation capability useful for teaching diagnostic reasoning. NEOMYCIN's rules are organized by a disease hierarchy and completely controlled by abstract metarules, constituting a diagnostic procedure; there is no backward chaining at the domain level. These metarules (currently 75 metarules organized into 40 subtasks) model both the hypothesis-oriented and the focused, forward reasoning of physicians.

By separating the diagnostic procedure from the disease knowledge, it is possible to provide explanations of strategy, regarding focusing and choice of questions when confirming a hypothesis. By stating the procedure abstractly, in a well-structured language, it is possible to provide abstract explanations. Such a general procedure can be run in a simulated mode to predict and describe student diagnostic behavior. Because the metarules model human reasoning, the advice supplied by the program might be usefully followed by a student.

GUIDON2 will be an intelligent tutoring system that uses the NEOMYCIN knowledge base and abstract diagnostic metarules as a source of teaching material The NEOMYCIN consultation system and explanation system are operational; a prototype modeling system (IMAGE) has been demonstrated. Recently, the metarules were reimplemented in MRS, using procedural attachment to access the EMYCIN domain knowledge.

Current projects include summarizing reasoning in explanations, evaluating the student model, formalizing alternative procedures followed by a student (enumerating bugs), and using INTERLISP-D to display knowledge structures. Research over the next two years may include incorporating the student model into explanations (reasoning about the rationale for an inquiry), converting the student modeling system into a belief system that reasons about its own interpretations (possibly with some learning capability), and developing an agenda-based tutoring capability.

While this is nominally a project focusing on intelligent tutoring, a parallel effort by Derek Sleeman will extend ideas on tutoring and student modeling in the domain of mathematics. An important motivation for the research is to develop better knowledge representations for expert systems We take seriously the fact that humans are at both ends of an expert system, as suppliers of expertise and as end users; thus, knowledge engineering must respect human ways of thinking—for improved performance, acceptability, and ease of system construction. NEOMYCIN's metarules can be used in other domains, with the practical benefit of making the next system easier to build and the scientific benefit of enabling us to test the generality of a formalized model of reasoning.

Personnel William J Clancey (contact), Diane Warner, David Wilkins, Derek Sleeman, Bruce Buchanan

References

Clancey, W. J (1983) The epistemology of a rule-based expert system: A framework for explanation. In *Proceedings of the National Conference on Artificial Intelligence, Washington, DC,* in press (Also, Rep STAN-CS-81-896 and Memo HPP-81-17, Computer Science Department, Stanford University.)

Clancey, W. J. (1983) Methodology for building an intelligent tutoring system In Kintsch, Polson, and Miller (Eds), *Methods and tactics in cognitive science* Hillsdale, NJ: Erlbaum, in press. (Also, Rep STAN-CS-81-894 and Memo HPP-81-18, Computer Science Department, Stanford University.)

London, B., and Clancey, W. J. (1982) Plan recognition strategies in student modeling: Prediction and description In *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA,* 335-338 (Also, Rep STAN-CS-82-909 and Memo HPP-82-7, Computer Science Department, Stanford University.)

Sleeman, D , and Brown, J S. (Eds) (1982) *Intelligent tutoring systems.* London: Academic Press.

The KBVLSI Project

The knowledge-based VLSI (KBVLSI) project is concerned generally with the nature of the design process and specifically with the design of integrated circuits. The primary focus of the project is the Palladio system, an exploratory environment for integrated-circuit design. Palladio is an environment for experimenting with design methodologies and knowledge-based design aids It differs from other prototype design environments by providing the means for constructing, testing, and incrementally modifying or augmenting design tools and design languages.

Palladio provides a testbed for investigating elements of circuit design including specification, simulation, and use of existing designs. It has facilities for conveniently defining models of circuit structure or circuit behavior These circuit models, called perspectives, are similar to levels of circuit design; the designer can use them interactively to create and refine circuit-design specifications. Perspectives can include composition rules that constrain how circuit components may be combined in that perspective to form more complex components. Palladio provides both an interactive graphics interface for displaying and editing structural perspectives of circuits in a uniform manner and a situation-action-rule, behavioral language with an associated interactive behavioral

editor for specifying a design from a behavioral perspective Further, a generic, event-driven simulator can simulate and verify the behavior of a circuit specified from any behavioral perspective and can perform hierarchical and mixed-perspective simulation Facilities are available for conveniently creating and using prototype libraries containing components of arbitrary complexity, specifiable from multiple perspectives.

Using Palladio, we are experimenting with numerous structural and behavioral perspectives that range from a sticks perspective (which specifies the actual planar topology of fabrication masks for the circuit) with an associated 3×4 level-strength logic behavioral perspective to a very abstract architectural perspective for investigating a novel pipelined MIMD machine.

We are developing knowledge-based design aids based on the perspective framework provided by Palladio. These design aids are small expert systems that perform some of the refinement necessary to move from an abstract circuit specification to more concrete specifications. For example, we have implemented an expert system design aid that assigns mask levels to the interconnect (wires and contacts) between components in a circuit specified in terms of switches and gates, and we are developing an expert system design aid to determine the gate sizes of the transistors in such circuit specifications that takes into account global speed and power goals and trade-offs.

Other research areas we are actively pursuing include: (a) the design of a language that spans the spectrum of functionality, behavior, and structure, thus eliminating some of the parallel specification languages currently required, and (b) the design of a language in which circuit-design problems (and theories of circuit design) can be stated, based on the assumption that the circuit-design problem and the circuit design co-evolve. Basic terms in such a language include design goals, tasks, constraints, and trade-offs.

Personnel Harold Brown (contact), Gordon Foyster, Christopher Tong, Jerry Yan

References

Brown, H., Tong, C , and Foyster, G. (1983) Palladio: An exploratory environment for IC design. Memo HPP-83-31, Heuristic Programming Project, Computer Science Department, Stanford University

The MOLGEN Project

The MOLGEN project is an eight-year collaborative effort among computer scientists and biologists to apply the methodologies of artificial intelligence to the domain of molecular biology. Our initial research goals were in the areas of knowledge representation and planning. This work led to the development of the Unit system (a frame-based system for knowledge representation, acquisition, and manipulation) and two expert systems that designed genetic experiments. One of the systems introduced the concept of "skeletal plans," abstracted outlines of experiment designs that are instantiated to specific experimental goals and environments. The other system introduced the concepts of "metaplanning," that is, making planning decisions in three planning "spaces" relating to overall strategy, domain-independent design decisions, and domain-dependent laboratory decisions, and "constraint propagation," that is, making no plan selection or ordering decisions until forced to by constraints from other plan steps.

Recent work has involved the synthesis of skeletal plans and metaplanning through the SPEX system SPEX is presently being used as part of a practical design system for the domain of cloning experiments We are currently adding capabilities for the diagnosis and debugging of experiment designs, determining the causes of failure when an experiment design is implemented in the laboratory and suggesting possible corrective actions.

We have proposed future work in a new direction, that of scientific theory construction and modification The goal is to emulate the problem-solving behavior of human scientists as they modify and extend existing scientific theories to predict and explain experimental evidence. Our initial approach to this problem will involve a blackboard architecture to capture the many different knowledge sources that appear to influence scientific theory formation; these include at least data-driven and theory-driven extensions, as well as various forms of analogic, abstract, and opportunistic reasoning.

Personnel· Peter E Friedland (contact), Douglas Brutlag, Laurence Kedes, René Bach, Peter Karp

References

Friedland, P (1979) Knowledge-based experiment design in molecular biology Rep. CS-79-771, Computer Science Department, Stanford University

Iwasaki, Y , and Friedland, P (1982) SPEX: A second-generation experiment design system. In *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA*, 341-344

Smith, R., and Friedland, P (1980) A user's guide to the Units system. Memo HPP-80-28, Heuristic Programming Project, Stanford University

Stefik, M. (1980) Planning with constraints Rep CS-80-784, Computer Science Department, Stanford University

The RX Project

The objective of the RX project is to develop methods for automatically discovering and confirming medical knowledge from large databases.

Our approach involves, first, building a knowledge base of medicine, statistics, epidemiologic study design, and methods of discovery and, second, using that knowledge base to automatically seek out and confirm medical relationships from the database.

A prototype system has been built including four major components: a knowledge base, a discovery module, a study module, and a statistical package interface. The system has run successfully on a subset of the American Rheumatism Association's database (ARAMIS), discovering and confirming knowledge that had not been previously well confirmed A key feature of the system is its ability to automatically incorporate newly confirmed causal relationships into a detailed machine-readable form. These causal relationships may be subsequently used in further iterations of the discovery cycle.

Our future plans include (a) redesigning the Discovery Module to include a greater repertoire of discovery methods, (b) increasing the generality and flexibility of the knowledge representation capabilities, and (c) increasing the generality of the Study Module.

Personnel Robert L Blum (contact), Gio C. M. Wiederhold, Mark Erlbaum, Michael G Walker

References

Blum, R. L. (1982) Discovery and representation of causal relationships from a large time-oriented clinical database: The RX project In D A. B. Lindberg and P. L Reichertz (eds), *Lecture notes in medical informatics* (Vol. 19). New York: Springer-Verlag.

Blum, R L (1982) Discovery, confirmation, and incorporation of causal relationships from a large time-oriented clinical database: The RX project. *Computers and Biomedical Research* 15(2):164-187

Blum, R. L. (1983) Representation of empirically derived causal relationships. *Proceedings 8th IJCAI. Karlsruhe, West Germany.* August 8-12, 1983

The IA Project

The goal of the IA project is the creation of an "intelligent agent" to assist users of operating-system utilities such as mail handling, file management, and document preparation. We envision a program that can accept a high-level statement of its user's goals, can synthesize and execute a sequence of operating-system commands to achieve those goals, and can deal with any resulting error conditions. In multicomputer communities such as the Arpanet, this can involve cooperation and competition among the agents on different hosts Ideally, the agent should be capable of sustained existence and thus be able to reroute mail, schedule its user's time, and handle other routine activities in its user's absence.

Funded by DARPA, the chief result of the work during the first two years is the creation of a mini intelligent agent (called IA) that handles file-movement problems between arbitrary sites in a multinetwork computer system. One thing that sets this program apart from more traditional operating system utilities is its ability to accept, utilize, and retain "rules" supplied by its user, for example, saying where all files of a certain type should be stored. A second key feature is its ability to handle errors due to disk-space limitations, host failures, time-outs, and so forth.

Although much of our current effort on the IA project is aimed at the maintenance and development of the IA program, the majority of the effort is devoted to basic research on problems of user interface, planning, and cooperative problem solving.

The key issues in user interface include language design and user modeling. A major user-interface subproject is directed toward the study of multiple-modality languages, including tables, charts, diagrams, color, and motion. The practical goal of this subproject is the creation of a program that takes as input a body of data and a set of communication goals and designs an appropriate display for that information.

The workhorse of a good intelligent agent is a planner capable of synthesizing a sequence of commands to achieve the users' goals. While much research on planning has already been done, there are significant problems in applying the technology to the operating system domain. One such problem is sensory planning, that is, the synthesis of plans to gather information. Another problem is execution monitoring. Due to the fallible nature of operating system commands (e.g., file transfer), it is often necessary to verify the success of one command before executing another. It is inefficient to put such checks after every command, but sometimes it is essential (e g., after a file transfer and before an expunge)

Multicomputer configurations pose both problems and opportunities for intelligent agents. In designing a plan to achieve a user's goal, the agent may have to rely on other hosts to achieve specific subgoals (e.g , in using a formatter or compiler not locally available). In another situation, it may decide to exploit slack time on remote hosts to gain higher throughput for its user (e.g., in a network of personal computers where some processors are idle). In either case, an agent must be able to decompose problems for distributed execution, monitor their execution, and recover in the event of processor or communication failure. It must take care to avoid deadlock, prevent loops, and synchronize activity properly. Of course, it may have to resolve conflicts between the goals of its local users and those of other hosts.

In addition to these mainline activities, there is a continuing empirical study of the ways in which humans use operating systems and an ongoing evaluation of the effectiveness of the IA program.

Personnel. Michael Genesereth (contact), Jeff Finger, Jeff Rosenschein, Jock Mackinlay, Vineet Singh

References

Finger, J. (1982) Sensory planning Memo HPP-82-12, Heuristic Programming Project, Computer Science Department, Stanford University.

Mackinlay, J (1983) Intelligent presentation of information: The generation problem of user interfaces Rep HPP-83-34, Heuris-

tic Programming Project Rep Computer Science Department, Stanford University

The MRS Project

The long-range goal of the MRS project is the construction of a program that can reason about and control its own problem-solving activity The program should be able to explain its actions as well as its results, and it should be able to accept advice from users who are unaware of its implementation The primary results of the research to date include a "metalevel" language adequate to describe the actions and goals of a problem-solving system, the development and formalization of several powerful domain-independent control strategies, and the implementation of a metalevel knowledge-representation system called MRS (for Metalevel Representation System).

MRS is intended as a practical tool for use by AI researchers in building expert systems. It offers a diverse repertory of commands for asserting and retrieving information, with various representations (e.g , property lists, propositional representation), various inference techniques (e g., backward chaining, forward chaining, and resolution, all with optional caching and truth maintenance), and various search strategies (e.g., depth-first, breadth-first, and best-first search). The initial system includes a vocabulary of concepts and facts about logic, sets, mappings, arithmetic, and procedures.

What differentiates MRS from many other knowledge-representation systems is its ability to reason about and control its own activity. In MRS, the system is treated as a domain in its own right. One can write sentences about subroutines and other sentences and allow the system to reason with them, just as it reasons about geology or medicine In practice, MRS uses this metalevel information in deciding how to satisfy its users' goals. Thus, one can switch representations or inference methods by changing these sentences, and one can easily implement a variety of control schemes.

As a practical knowledge-representation tool, MRS has been used in the construction of a variety of expert systems, including an automated diagnostician for computer hardware faults (DART), a calculus program (MINIMA), a simulator for digital hardware (SHAM), a mini tax consultant (IDWONNA), and an implementation of the NEOMYCIN infectious-disease diagnosis and tutoring system

In addition, the system has served as a testbed for basic research in metalevel reasoning and control. One result of this research has been the development of a metalevel language that allows partial specifications of program behavior and thereby facilitates incremental system development and the integration of disparate architectures such as demons, object-oriented programming, and controlled deduction. A second result is an analysis of conjunctive problem solving and a mini expert system that decides the ideal order in which to work on conjuncts A third result is a mini expert system that controls the system's activity in computing all the solutions to a problem.

Current work on the project is proceeding in three primary directions. The first is the continued study of run-time control issues and the development and formalization of powerful domain-independent control techniques like the conjunct-ordering heuristics mentioned above. The second and more recent direction is the construction of a "system-building consultant" that operates at "build time." Given a knowledge base, such a system would give advice on the most appropriate representations, inference methods, search strategies, and so forth. It would also be able to compile rule sets into more efficient code, while taking into account the control suggestions of the system builder. Finally, there is a significant commitment to maintaining and developing MRS as a practical knowledge-representation tool, with an emphasis on the use of interactive graphics in the creation of debugging tools for reasoning systems.

Personnel: Michael Genesereth (contact), Russell Greiner, David E. Smith, Richard Treitel

References

Genesereth, M. R (1983) A meta-level representation system Rep HPP-83-28, Heuristic Programming Project, Computer Science Department, Stanford University

Genesereth, M R (1983) An overview of meta-level architectures In *Proceedings of the National Conference on Artificial Intelligence, Washington, DC,* in press

The Blackboard Architecture Project

The blackboard architecture is a problem-solving framework developed for the Hearsay-II speech-understanding system. Since its development, the architecture has been exploited in a variety of AI systems and simulations of human problem solving. These systems incorporate many different inference methods and search strategies They attack problems ranging from sonar signal interpretation to planning the movement of vehicle fleets. Despite the blackboard architecture's apparent power and generality, our understanding of it remains informal and fragmented. This project is intended to integrate what is currently known about the blackboard architecture and to develop, analyze, and evaluate a comprehensive blackboard model for problem solving

Taking Hearsay-II as its prototypes, the project defines the blackboard architecture in terms of three features First, independent knowledge sources generate, modify, and relate solution elements (e.g., hypotheses, decisions) Knowledge sources have a condition-action format and are event-driven They may embody top-down, bottom-up, or other inference mechanisms. Second, a structured, global database called the blackboard mediates all knowledge-source interactions Solution elements on the blackboard trigger knowledge sources

whose actions record new solution elements on the blackboard. The blackboard structure represents solution elements at multiple levels of abstraction and for different intervals of the solution. Third, an agenda-based scheduler determines which currently triggered knowledge sources should execute their actions at each point in the problem-solving process. The scheduler bases its decisions on knowledge-source characteristics, triggering-event characteristics, potential-action characteristics, current blackboard contents, and so forth.

As defined, the blackboard architecture exhibits a problem-solving style that is characteristically incremental and opportunistic. Partial solution "islands" emerge, one element at a time, in different structural partitions of the blackboard. New solution islands appear and existing islands grow wherever the opportunities are most promising. Eventually, mutually supportive partial solutions merge to form a complete solution.

In the area of control, the project is developing a control model that can accommodate the several mechanisms employed in previous applications. The control model is, itself, framed in the blackboard architecture. Scheduling and strategic planning are conducted by control knowledge sources operating on a control blackboard In the area of parallel computation, the project is developing a variation on the control model that can coordinate the activities of multiple processors. The model is designed to exploit processors differing in computational power, to enable the system to allocate its processors efficiently and asynchronously, and to degrade gracefully upon processor failure.

The project will evaluate the performance characteristics of the synthesized models in a travel-planning domain and perhaps a signal-interpretation domain.

Personnel: Barbara Hayes-Roth (contact), Edward A. Feigenbaum, Penny Nii

References

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

The GLISP Project

GLISP is a programming language that is compiled into LISP The GLISP language allows programs to be written in terms of objects and their properties and behavior; the GLISP compiler, in turn, converts such programs into efficient programs in terms of the implementations of objects in LISP.

GLISP provides mechanisms by which the implementation of an object and of operations on that object can be described, thus allowing programs to be written in terms of objects rather than in terms of the implementations of objects. This is similar in concept to object-centered programs which are slower than ordinary programs; GLISP, however converts programs written in object-centered form into conventional programs that run efficiently.

Programmers need to be able to examine the data their programs use. The GEV window-based data-inspection program displays data according to its GLISP description. The user can display computed data, send messages to objects, or "zoom in" on data of interest to display it in greater detail Looping programs that operate on the data currently being inspected can be created interactively by menu selection.

GLISP and GEV provide an integrated programming environment with facilities for automatic compilation, interpretive execution, and window-based inspection of data. GLISP and GEV are available for the major dialects of LISP.

GLISP programs involve three kinds of specifications object descriptions, generic programs, and application programs. Object descriptions describe the data structures that embody the states of objects, properties of objects that are computed rather than being stored, messages that can be sent to objects to cause them to perform actions, and classes to which objects belong. Generic programs express operations on abstract classes of objects and can be instantiated for particular data types as programs that are specialized to deal with those types. Application programs in GLISP are much like those in traditional programming languages however, GLISP programs can be written in terms of the objects and operations of the problem domain rather than in terms of the LISP base language.

A compiler for the GLISP language has been implemented and has been released to outside users. The GEV data inspector, written in GLISP, has also been released GLISP is currently being used for implementation of knowledge-based systems at HPP and at other AI research centers Current research is focused on combining separate generic algorithms into unified programs.

Personnel: Gordon S. Novak, Jr. (contact)

References

Novak, G. S., Jr. (1982) Data abstraction in GLISP. Memo HPP-82-34, Heuristic Programming Project, Computer Science Department, Stanford University. (To appear in *Proceedings of SIGPLAN '83 Symposium on Programming Language Issues in Software Systems*)

Novak, G. S., Jr. (1982) GLISP: A high-level language for AI programming In *Proceedings of the Second National Conference on Artificial Intelligence, Pittsburgh, PA*, 238. (Also available as Memo HPP-82-35, Heuristic Programming Project, Computer Science Department, Stanford University.)

Novak, G S., Jr. (1982, 1983) GLISP user's manual. Memo HPP-82-1, Heuristic Programming Project, Computer Science Department, Stanford University.

The AGE Project

The objective of the AGE (Attempt to GEneralize) project was to design a software laboratory for building knowledge-based application programs. It attempted to define and accumulate knowledge-engineering tools, with rules to guide in the use of these tools. The first system, AGE-1, has been completed and is available The design and implementation of this system were based primarily on the experiences gained in building knowledge-based programs within HPP in the last decade. It contains a collection of tools for constructing application programs suited to blackboard formulation (as in HASP and CRYSALIS) and backward-chained rules (as in MYCIN). In addition, AGE-1 has facilities to aid the user in the construction, debugging, and running of his or her program.

AGE-1 has now been used to build several applications, both within HPP and in industry. Our experiences in the construction of the system and in its use remain to be documented. A set of specifications for AGE-2 (a next-generation system that remedies some of the shortcomings of AGE-1 and extends its capabilities) has been drawn up, but it will not be implemented. Instead, the AGE efforts will be merged with the Advanced Architectures project and will focus on the issues of concurrent processing for the blackboard programming framework.

We have therefore reached the successful conclusion of the AGE Project. We hope to have a retrospective published within a year.

Personnel Penny Nu (contact), Nelleke Aiello

References

Nii, P (1980) An introduction to knowledge engineering, blackboard model, and AGE. Memo HPP-80-20, Heuristic Programming Project, Computer Science Department, Stanford University.

Aiello, N , Bock, C , Nii, H P , and White, W C (1982) The joy of AGE-ing: An introduction to the AGE-1 system. Memo HPP-81-23, Heuristic Programming Project, Computer Science Department, Stanford University

Aiello, N (1983) A comparative study of control strategies for expert systems: AGE implementation of three variations of PUFF In *Proceedings of the National Conference on Artifical Intelligence, Washington DC*, in press

The HPP Computing Environment

The current HPP computing resources are a networked mixture of mainframe host computers, LISP workstations, and network utility servers, reflecting the evolving hardware technology available for AI research Our host machines include a DEC 2060 and 2020 running TOPS-20 (these are the core of the national SUMEX biomedical computing resource) and a VAX 11/780 running UNIX. In addition, we have five Xerox 1100s (Dolphins) and a Symbolics LM-2 LISP machine. Network printing, file, gateway, and terminal interface services are provided by dedicated machines ranging from VAX 11/750s to microprocessor systems. These facilities are integrated with other computer-science resources at Stanford through an extensive ETHERNET and to external resources through the ARPANET and TYMNET. The current resources are inadequate to support a 70-person project and we expect to acquire additional Xerox 1108 (Dandelion), 1132 (Dorado), Symbolics 3600, and other appropriate LISP workstations over the next few years.

Funding Acknowledgments

HPP research has been funded from many sources over the years, currently including the Defense Advanced Research Projects Agency (contracts MDA 903-80-C-0107, N000 14-81-K-0303, and N000 39-83-C-0136), the Department of Health and Human Services (NIH grants RR-00612, RR-00785, and RR-01631 [pending]; NLM grants LM-03370 and LM-03395; and NCHSR grant HS-04389), the Office of Naval Research (contracts N000 14-79-C-0302, N000 14-81-K-0004, and N000 14-80-C-0609), the National Science Foundation (grants ECS-80-16247 and MCS-81-177330), the National Aeronautics and Space Administration (contracts NAG-5261 and NCC-2-220), International Business Machines, and Schlumberger