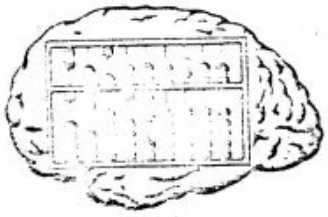
SIGART SPECIAL ISSUE ON MACHINE LEARNING

by

Tom M. Mitchell
Jaime G. Carbonell
Ryszard S. Michalski (Editors)

SIGART No. 76, April, 1981.

SIGART NEWSLETTER



A Quarterly Publication of the ACM Special Interest Group on Artificial Intelligence

No. 76 April 1981

SIGART Special Issue on Machine Learning

Editors' Introduction to the Issue

Current research on Machine Learning encompasses a diverse set of approaches, and of opinions regarding where the important issues lie. The significant increase of interest and research activity in Machine Learning over the past few years has led us to organize this special issue of SIGART, whose purpose is to provide a snapshot of current research in this field. This issue contains a set of summaries of ongoing research, solicited from the community at large, and received from thirty-five research groups from around the world. A substantial bibliography on Machine Learning is also included.

Research summaries were obtained in response to our general call for contributions, which appeared in two previous SIGART issues. All contributions that were submitted have been included. Our editing policy has been to avoid evaluating the work reported here, and to let the authors speak for themselves. At the same time, we wanted to impose enough syntactic structure on the issue to allow easy digestion of the material. Therefore, some changes have been made to the format of summaries, in order to organize each according to Topic, Personnel, Objectives and Methods, Results, Future Plans, Support, and References. The content of the summaries has been left unchanged.

In addition to the research summaries, this issue contains a substantial bibliography on Machine Learning, compiled by Bernard Nudel and Paul Utgoff from their own entries as well as those provided by several others. Compilation of this bibliography has been a monumental task, and we greatly appreciate the work that Paul and Bernard have put into its preparation.

A brief topic index and an author index into the research summaries follow this introduction. The research summaries are then listed in alphabetical order by the last name of the first author. The bibliography on Machine Learning follows these summaries. We have found the research summaries useful in providing succinct characterizations of basic objectives, approaches, and results, while providing references to more detailed descriptions of the work. (We are currently editing a book on Machine Learning, scheduled to appear in late 1981, which will contain detailed descriptions of several of the research projects reported here, as well as other work in this field.)

Our thanks to the people who wrote this issue: all those who contributed summaries of their research. Cathy Dolese and Chris Loungo have provided clerical support in the preparation of this issue. Without them, the issue would certainly not have made it to press by the deadline. Our thanks to them for their help.

Tom Mitchell
Jaime Carbonell
Ryszard Michalski

March, 1981

Topic Index to Research Summaries

The following is a brief topic index into the research summaries that follow, with each summary referenced by the last name of the first author. The summaries are indexed along three dimensions: Representation of learned knowledge, Application area, and Main goal of the research. We have only classified those research summaries for which we felt the correct classification was fairly apparent. As with any attempt at classification of this work, ours is unavoidably approximate and incomplete. We apologize in advance to authors for all errors and omissions.

Representation of learned knowledge:

Induction of Grammars (Grammatical Inference):
[Angluin], [Berwick], [Harrison].

Induction of Structural and Logical Discriminants (Concept Formation, Learning structural descriptions):
[Banerji], [Chisholm], [Freuder], [Herman], [Michalski], [Srinivasan].

Induction of Control Strategies, Heuristics, Problem-Solving skills:
[Carbonell], [CohenD], [DeJong], [Findler], [Harrison], [Hayes-Roth], [Holland], [Kibler], [Mitchell], [Neches], [Novak], [Mostow], [Rychener], [Schank], [VanLehn].

Induction of Programs (Automatic Programming):
[Andreae], [CohenB], [CohenD].

Expansion and Reorganization of Episodic Memory:
[Carbonell], [Schank].

Application Areas (where discernible):

Mathematics:
[Mitchell], [Neches], [Sleeman], [VanLehn].

Natural Language:
[Berwick], [Carbonell], [Haas], [Schank], [Whitehill].

Physics:
[Langley], [Novak].

Plant Pathology:
[Michalski].

Student Modelling:
[Clancey], [Sleeman], [VanLehn].

Theorem Proving:
[CohenD], [Srinivasan].

Vision:
[Herman].

Main Goal of the Research:

Adaptive Networks Research:
[Klopf].

Knowledge Acquisition for Expert Systems:
[Chisholm], [Haas], [Harrison], [Hayes-Roth], [Kibler], [Michalski], [Mitchell], [Mostow], [Rychener].

Modeling Human Learning:
[Berwick], [Carbonell], [Clancey], [DeJong], [Holland], [Neches], [Novak], [Schank], [Sleeman], [VanLehn].

Research on systems that define new terms to extend their description language:
[Langley].

Theoretical and Formal Models of Induction:
[Angluin], [Banerji], [Findler], [Harris], [Kugel], [Michalski], [Mitchell], [Shapiro], [Srinivasan].

Computational Complexity of Induction:
[Angluin], [Banerji].

INDEX OF RESEARCHERS

The following index of researchers lists all people mentioned as participants in the research summaries presented here. Associated with each author is a pointer to the appropriate research summary (i.e., last name of first author).

[Andreae] CURRENT RESEARCH IN MACHINE LEARNING

Personnel: Dr. John H. Andreae (Reader). Patrick L. Chitaka, Bruce A. MacDonald, Neil L. Scott, Trevor J. S. Wignall. (Department of Electrical Engineering, University of Canterbury, Christchurch, N.Z.)

Objectives and Methods:

Our objective is to increase the understanding of human intelligence by attempting to simulate and construct an intelligent robot.

It is a basic postulate of our work that intelligence is learned. We give our "multiple context learning system", called PURR-PUSS, the minimum of priming and we avoid giving it task-specific capabilities except in so far as it is designed to exploit the robot "body" in which it functions. The second basic postulate is that intelligence implies self-motivation. The system must set its own goals and seek them.

The learning system is a very general form of production system, constrained to acquire productions only from its own behavior. Starting with no productions in its long-term memory, the system has literally no knowledge about its "body" or environment. Activity is initiated by built-in reflexes and by forced actions (e.g. by a teacher in the environment). The motivation of the system is achieved by the automatic formation of "novelty goals", which are events occurring in a context for the first time. The short-term memory, which comprises a set of productions derived from the current behavior, is called a "multiple context" and is a parallel processing system enabling prediction and decision. Computer versions of the system employ hashing of contexts for the storage and retrieval of productions in long-term memory, so the system can be given an indefinitely large memory without the speed of interaction deteriorating.

Results:

The system has been shown to be teachable with a variety of simple tasks. It can learn on its own without a teacher and it has Turing Machine power. Generalization and discrimination are basic to its operation. The system has been connected to several small robots. A neurochemical version has been proposed and simulated.

Future Plans:

A robot is being constructed from a motorized wheelchair. This is being given simple forms of hand, vision and speech. We plan to increase the limited learning period possible with existing memory by adding a microprocessor to address megabytes of random access memory. Disk memory is too slow. We shall continue to search for a useful formal way of describing the complex data structures set up by the multiple context.

Support:

The work is carried out with a small grant from the New Zealand Ministry of Defence.

References:

The work has been fully documented in the series of reports entitled "Man-Machine Studies, reports numbered UC-DSE/1(1972) to UC-DSE/16(1980), ISSN 0110 1188, edited by J. H. Andreae" and issued from the Department of Electrical Engineering, University of Canterbury, Christchurch, New Zealand.

Andreae, J. H., Cleary, J. G. (1976): A New Mechanism for a Brain. *Int. J. Man-Machine Studies 8* (1) 89-119.

Andreae, J. H., (1977): *Thinking with the Teachable Machine.* Academic Press.

Andreae, J. H. (1978): PURR-PUSS: Computer Simulation of a Teachable Machine. *SAGSET J. 8* (4) 123-133.

Andreae, P. M., Andreae, J. H. (1978): A Teachable Machine in the Real World. *Int. J. Man-Machine Studies 10* (3) 301-312.

Andreae, J. H. & Andreae, P. M. (1979): Machine Learning with a Multiple Context. *Proc. IEEE Conf. on Cybernetics and Society.* Denver, Colorado. October. 734-739.

Cleary, J. G. (1980): *An Associative and Impressible Computer.* Ph.D. Thesis. University of Canterbury, Christchurch, New Zealand.

[Angluin] INDUCTIVE INFERENCE: EFFICIENT ALGORITHMS

Personnel: Dana Angluin, (Computer Science Department, 10 Hillhouse Ave., Box 2158 Yale Station, New Haven, CT 06520)

Objectives, methods, and results:

The goal of this research is to achieve an understanding of the mathematical and structural properties of various types of rules that are readily inferable from examples, and to develop efficient algorithms to perform such inferences. Incorporating simple, reliable, well-understood induction strategies into computer programs would make them much easier to use, particularly by non-experts. Unfortunately, the theory underlying any such development is largely non-existent at present.

Starting from a basis in theoretical computer science (formal language theory, recursive function theory, complexity theory, and analysis of algorithms) and the pioneering work of Gold, Feldman, and the Blums on formalizing an abstract theory of inductive inference, I have begun to develop a body of concepts, techniques, and results for the analysis of inductive inference in specific, concrete domains. In [1] there are results indicating the computational intractability of finding the smallest finite automaton or regular expression compatible with given positive and negative examples of a regular set, even under various restrictions on the data and the form of the answer. In [2] there is a characterization of those families of recursive languages identifiable in the limit from positive data. In [3] there is an efficient algorithm to infer patterns like 4xx43 from data like {41143, 4000043, 4141443}. In [4] there is another efficient inference algorithm, to infer reversible regular sets from positive data. For example, this algorithm infers the set of all strings containing an even number of 0's and an even number of 1's from the data {00, 11, 0000, 0011, 0101, 0110, 1001}.

The work of Ehud Shapiro, who is a graduate student in Computer Science at Yale, concentrates on refining and specializing general algorithms for inferring logical theories from facts. He has been experimenting with implementations of several different general algorithms in PROLOG. These algorithms are theoretically complete in the sense of the Blums' work, and also may lead to practical realizations. He has found a general method for resolving contradictions in the currently hypothesized theory, which formalizes a notion of crucial experiments. A technical report on this work will soon be available [5].

Carl Smith, of the Computer Science Department at Purdue, and I are currently writing a survey article on general and specific theoretical results in inductive inference. Carl has already prepared a bibliography on this area, which he is sending to SIGART.

My future work will be along these same lines, with perhaps more emphasis on finding novel applications for the algorithms developed.

Support:

The support of the NSF under grant number MCS 8002447 for this vork is gratefully acknowledged.

References:

[1] D. Angluin. On the complexity of minimum inference of regular sets. *Inform. Contr.* 39:337-350, 1978.

[2] D. Angluin. Inductive inference of formal languages from positive data. *Inform. Contr.*, to appear.

[3] D. Angluin. Finding patterns common to a set of strings. *J. Comp. Sys. Sci.*, to appear.

[4] D. Angluin. Inference of reversible languages. Submitted for publication.

[5] E. Shapiro. Inductive inference of first order theories from facts, preliminary report. To appear as a Yale Computer Science Dept. technical report.

[Banerji] RESEARCH ON LEARNING IN A FLEXIBLE DESCRIPTION LANGUAGE

Personnel: Ranan Banerji (Temple), Brian Cohen, Claude Sammut (Univ. New South Wales), Tom Mitchell (Rutgers), Yves Kodratoff, Gerard Guiho (Univ. of Paris).

Objectives and methods:

The relationship between flexibility of description language and efficiency of learning is being studied in two applications areas: learning heuristics from experience and learning Lisp programs from input-output pairs. It is believed that the efficiency of learning as well as the complexity of the concept learned is dependent on the richness of the description language, i.e. the number of statements about the examples one can deduce from the input statements. However, the efficiency-richness curve has a maximum point, since the efficiency of deduction falls as the language gets richer. At the maximum, the language may be said to be tailored to the task.

The purpose of the project is to study this relationship by embedding the tailored language into a flexible language where the repertory of deducible sentences can be expanded by definition and by learning.

Results:

We have translated the expressions used in the description language of the LEX system (1,2) (used for learning heuristics for formal integration) into the DL language (in the class discussed by Banerji, Cohen and Sammut (3,4,5,6). The LEX matcher essentially leads the system to specific deductions ("if fr1 is SIN then it is TRIG and then it is TRANSCENDENTAL"). In Cohen's CONFUCIUS or in the later systems, the system would find these deductions efficiently if TRIG or TRANS were the only predefined predicates pointed to by SIN but would fall in efficiency if there were other related predefined predicates, i.e. the efficiency of LEX can be looked upon as a result of restricting its knowledge.

Future Plans:

Since it is found that the LEX language is in need of expansion there is the problem of expanding it "to fit the problem". The relationship between the technique developed in LEX will be compared to the technique of "expansion and relaxation" used by Cohen. A similar analysis is planned for the comparison of DL (as the embedding language) with the restricted LISP used by Kodratoff (7) for learning languages from input-output pairs.

References:

(These references are to background material. No publications on the project has yet seen the light of day.)

(1) Mitchell, T.M. and Utgoff, P.E.,"Improving Problem Solving Strategies by Experimentation: A Proposal" Rutgers University Computer Science Technical Report CBM-TR-106, Nov. 1979

(2) Mitchell, T.M., Utgoff, P.E. and Banerji, R.B.,"Learnin Problem Solving Heuristics by Experimentation", Proceedings of th Workshop on Machine Learning, CMU, June 1980.

(3) Cohen, B.L.,"A Powerful and Efficient Patter Recognition System", Artificial Intelligence, 9, p.223 (1977)

(4) Cohen, B. L. and Sammut, C. E.,"Pattern Recognition ar Learning with a Structural Description Language", Proceedings the 4th International Joint Conference on Patter Recognition,(Kyoto, 1978)p.394

(5) Banerji, R. B.,"Using a Description Language as Programming Language", Proceedings of the 4th International Joi Conference on Pattern Recognition (Kyoto 1978),p.346

(6) Cohen, B.L., "Program Synthesis Through Conce Learning", Proceedings of the International Workshop on Progra Construction, (Chateau de Bonas, France. September 1980)

(7) Kodratoff, Y., "A Class of Functions Synthesized from Finite Number of Examples and a LISP Program Schem International Journal of Computer and Information Sciences, p.489 (1979).

[Berwick] THE ACQUISITION OF SYNTACTIC KNOWLEDGE: LEARNING STRUCTURAL DESCRIPTIONS OF GRAMMAR RULES FROM EXAMPLE

Personnel: Robert C. Berwick

Objectives and Methods:

A principal goal of modern linguistics is to account for apparently rapid and uniform acquisition of syntactic knowled given the relatively impoverished input that evidently serves as basis for the induction of that knowledge -- the so-called "project problem." At least since Chomsky, the usual response to projection problem has been to characterize knowledge of langu as a grammar, and then proceed by restricting so severely the cl of grammars available for acquisition that the induction task greatly simplified -- perhaps trivialized.

The research summarized here describes an implemen LISP program that explicitly reproduces this methodolog approach to acquisition -- but in a computational setting. It a what constraints on a computational system are required to ens the acquisition of syntactic knowledge, given relatively plaus restrictions on input examples (only positive data of limi complexity). The linguistic approach requires as the output acquisition a representation of adult knowledge in the form grammar. In this research, an existing parser for English, Mar PARSIFAL [1980], acts as the grammar. PARSIFAL divides ne into two parts: an interpreter and the grammar rules that interpreter executes. We mimic the acquisition process by fixir stripped-down version of the PARSIFAL interpreter, ther assuming an initial set of abilities (the basic PARSIFAL structures, a lexicon, and two simple phrase structure schem The simple pattern-action grammar rules and expansions of the phrase structure rules are acquired, on the basis of induction f grammatical sentences with a degree of embedding or two or Note that the program is deliberately restricted to draw inferer from positive-only evidence; no negative reinforcement is provid

Results:

To date, the accomplishments of the research are two-f First, from an engineering standpoint, the program succ

admirably; starting without any grammar rules, the currently implemented learning version of PARSIFAL (dubbed LPARSIFAL) acquires from positive examples many of the rules in a "core grammar" of English originally written by Marcus. The currently acquired rules handle simple declaratives, auxiliary verb inversion, simple passives, simple wh-movement, topicalization, imperatives, and negative adverbial preposing.

More recently, there has been some progress towards getting the procedure to acquire the context-free base rules presumably a part of an adequate grammar of English. These rules include the basic word-order expansion rules of a language, e.g., S-->Noun Phrase Verb Phrase, in English. This effort is based upon the X-bar theory of Chomsky and Jackendoff [1977], whereby the context-free base rules of all languages are presumed to be derived from just a few "skeleton" rules of a particularly simple sort. For instance, the basic sentence expansion rule can be considered to be of the form, S-->NP VP or S-->VP NP; exposure to positive example sentences tells the acquisition procedure which expansion order is to be selected.

But there is a second, more important accomplishment of the research. To ease the computational burden of acquisition it was found necessary to place certain constraints on grammar rule application and on rule form. The constraints on rule application can be formulated as specific locality principles that govern the operation of the parser and learning procedure. The LPARSIFAL constraints appear to be computational analogues of the restrictions on a transformational system advanced by Wexler and Culicover [1980]. In their independent but related formal mathematical modelling, they have proved that these restrictions suffice to ensure the learnability of a transformational grammar, a fact that might be taken as independent support for the basic design of LPARSIFAL.

Future Plans:

(1) Expand the work on the acquisition of base context-free rules (2) Investigate the acquisition of lexical entries. (3) Establish formal results about the connections between the Wexler-Culicover formal learnability results and the LPARSIFAL constraints.

Support: This work is funded under the MIT-AI lab's ARPA and ONR contracts.

References:

Berwick, R. "Computational Analogues of Constraints on Grammars," Proceedings of the 1980 Annual Meeting of the Association for Computational Linguistics.

Berwick, R. "Learning Structural Descriptions of Grammar Rules from Examples," unpublished MIT MS thesis, 1980. (Revised version to appear as MIT TR-578)

Berwick, R. "Learning Structural Descriptions of Grammar Rules from Examples," IJCAI 79, Tokyo, Japan.

Marcus, "A Theory of Syntactic Recognition for Natural Language," Cambridge, MA: MIT Press, 1980.

[Carbonell] MACHINE LEARNING RESEARCH SUMMARY

Personnel: Jaime G. Carbonell and Peter Neuss. Carnegie-Mellon University, Computer Science Department.

Objectives, methods, and results:

THE META PROJECT

META is a Natural Language learning system whose specific objective is to acquire new word definitions and new concepts from contextual information in interactive dialogues. It is an instance of a learning-from-examples method, with a difference: learning proceeds in a reactive, knowledge-rich environment. Our initial research indicates that the interactive nature of the environment ought to be a crucial component of any general learning system. In brief, our investigations have led us to formulate the following hypotheses, which we intend to test and pursue further in the immediate future:

(1) Learning requires progressive refinement -- It is unreasonable to expect that a computer system (or a human) learn a concept or a skill without error, in its full embellished form, in one brief learning session. It must undergo a sequence of progressive test-and-update stages. In other words, a concept can be learned by first inducing a rough approximation of its final form and successively correcting this approximation with more accurate, more detailed information.

(2) Interaction with a reactive environment -- Interaction is the engine that drives learning processes. A learner must be able to direct queries to its teacher or perform experiments on its environment. It must be able test out new concepts and skills as it learns them.

(3) Reasoning by Analogy -- The more a system can learn by relating new concepts to old, by modifying existing concepts, or using chunks of existing concepts as building blocks, the more robust and general its learning mechanisms will be.

META is a vehicle for investigating these hypotheses. We are currently in the process of developing parts of the computer implementation.

INTERACTIVE CONCEPT ACQUISITION ON HIERARCHICAL MEMORY STRUCTURES

Learning does not occur in the absence of other cognitive demands. This project is focused in part on viewing learning as an essential component of understanding, problem solving and memory organization, and in part on the need to extend an interrelated memory model -- rather than simply acquiring a self-contained, disembodied "concept".

Our initial computational realization of an extensible memory model is based on a hierarchical-inclusion memory organization and an active learner interacting with a reactive environment. The program exploits the inclusion hierarchy of a semantic network to guide the classification and identification of new concepts. Example generation is developed as an indispensable tool to expedite the convergence of a best-first search on the proper concept description. In addition, generating and testing new examples of a learned concept adds a measure of robustness, as the system is able to test effectively its newly acquired knowledge. Hence, this project also tests the hypotheses developed in the formulation of the META project. Finally, we hope that our episodic and semantic memory-extension model is applicable to the acquisition and refinement of new procedural skills.

Support:

In addition to Jaime Carbonell and Peter Neuss, working on both projects, Greg Hood started work on natural language learning and Monica Lam started work on reasoning and learning by analogy.

Our primary funding source is the Office of Naval Research (Grant number N00014-79-C-0661), which is also supporting other Learning research in Psychology and Computer Science at CMU. ONR also supported the CMU workshop- symposium on Machine Learning last summer. We gratefully acknowledge their generous

support.

References:

1. Carbonell, J. G., "Towards a Self-Extending Parser," Proceedings of the 17th Meeting of the Association for Computational Linguistics, 1979 , pp. 3-7.

2. Carbonell, J. G., "Metaphor - A Key to Extensible Semantic Analysis," Proceedings of the 18th Meeting of the Association for Computational Linguistics, 1980 .

3. Carbonell, J. G., "Δ-MIN: A Search-Control Method for Information-Gathering Problems," Proceedings of the First AAAI Conference, August 1980.

[Chisholm] AN AIDE FOR THEORY FORMATION

Personnel: Ian Chisholm, Derek Sleeman. University of Leeds, UK. (Ian Chisholm is now at Aston University, Birmingham, UK.)

Objectives, methods, and results:

The central problem addressed by this work is the acquisition of knowledge from experts, and as such it is closely related to the MYCIN and TIERESIAS systems. In our system, the domain 'theory' is formulated as a series of production rules; the expert/user then presents the system with problems and the system will, if the rules permit, solve these problems. If desired the expert can then interrogate the system as to how the problem has been solved and can, using editing facilities analogous to MYCIN's modify the ruleset.

In addition, we have added facilities which 'observe' certain kinds of regularities in the examples provided by the expert. That is, in Stanford's terminology, we have provided several modes which can perform model-directed inference which operate on the production rules and additional domain-knowledge provided in a semantic net. These facilities include the ability to resolve conflicts, produce analogies and carry out generalizations of features in the examples provided. We note that both the latter facilities rely on the ability to recognize the pertinent features of a problem and hence are closely related. The system uses the Semantic Net to suggest new concepts for the Condition part of a rule; for example having seen instances of a CHLORO-X and a BROMO-X behaving similarly, under certain conditions, it gives the user the choice between the BROMO-CHLORO concept and that of Halogen (the superconcept).

These inference modes make the current system a valuable Aide for Theory formation, capable of active participation in checking the completeness and consistency of the domain-knowledge/theory. (For this reason we have named the system CONCHE, CONsistency CHEcker [1]). Although CONCHE has been developed in the context of acquiring expert knowledge, we suggest that many of these capabilities would be valuable in other contexts, for example in ICAI systems. Collins has noted that human tutors make analogies with more familiar situations when faced with new situations. A teaching system incorporating some of these ideas could reason by analogy when operating with incomplete knowledge.

References:

1. I.H. Chisholm & D.H. Sleeman, 1979, An Aide for Theory Formation in "Expert Systems in the Micro-Electronic Age" ed D. Michie, EUP, pp. 202-212.

[Clancey] GUIDON: EXPLORATION OF TEACHING AND DIAGNOSTIC STRATEGIES

Personnel: William J. Clancey, Bruce Buchanan, Robert London, Reed Letsinger, Paula McKenzie, Tim Beckett, MD (consultant). Heuristic Programming Project, Stanford University

Objectives, methods, and results:

The purpose of this research is to develop a knowledge-base tutorial program for diagnosis problems in medicine. In order t teach diagnostic strategies, we assume that the program must buil a model of the student's problem-solving approach as he collect data and makes hypotheses. That is, the program must learn abou the student's plans, as well as his knowledge of data/hypothesi associations. GUIDON's student model has four ordere components:

1. Can the student USE the program? I.e., is he able to enter recognizable input?
2. Is the dialogue with the student COHERENT? I.e., are there recognizable patterns of student input and meaningful transitions between segments of behavior?
3. Is the student PASSIVE OR ACTIVE? I.e., does he use his own knowledge to solve the problem, or does he rely on the tutor's initiative and ability to provide help?
4. Does the student have a STRATEGY for solving the problem? I.e., is there some plan that organizes the student's data measurements and hypothesis selection?

The model is designed so each level of analysis determir whether more costly recognition routines should be called into pl Thus, if the student's behavior is coherent, but passive, it does make sense to ask what strategy he is using for generat hypotheses.

Student initiative and response to the tutorial progr constitute the low level observations that GUIDON abstracts int belief model which can be used to generate problem solving ad (perhaps at the level of using the program itself effectively). interaction bears some resemblance to that in PARRY where bel are accumulated over time based on the speaker's intent. F simple patterns map the 20+ options for student input into a categories of initiative, for example, "requesting assistance" "collecting data." Second, "segment recognizers" signal that the a low level pattern of behavior, such as "the student is askir series of questions directed towards one hypothesis" or "the stu is trying to determine the relevance of a subgoal." Third, "f shift patterns," inspired by PARRY's design, relate student beha to the program's remarks, for example, "following up on a hin "pursuing details." This much of the student model is now t tested.

To recognize student problem-solving approaches (phas we are re-implementing MYCIN's meningitis knowledge, so the rules are controlled by a set of general meta-rules for diagr hypothesis formation. In the current design, these situation/a rules suggest plausible diagnostic actions--shifting focus, colle data, reviewing the differential, posing an hypothesis, etc. Ou is to extend the current model of expert behavior to incorp common suboptimal approaches. Until we better understan variety of student behavior, we will follow the ennume approach, as in Brown's BUGGY program.

In summary, GUIDON's learning consists of simple clus and pattern analysis of abstracted observations. This analysis particular interest because of its combination of dialogue inter knowledge (the segment recognizers and focus shift patterns) model of hypothesis formation in diagnostic problem-solving.

Support:

This research is funded jointly by ARPA and ONR (Contract N-00014-79-C-0302).

References:

Clancey, W.J. Tutoring rules for guiding a case method dialogue. Int J of Man-Machine Studies, 1979, 11, 25-49.

Clancey, W.J. Dialogue Management for Rule-based Tutorials. Proceedings of Sixth IJCAI, 1979.

Clancey, W.J., Shortliffe, E.H., and Buchanan, B.G. Intelligent computer-aided instruction for medical diagnosis. Proceedings of the Third Annual Symposium on Computer Applications in Medical Care, Silver Spring, Maryland, October 1979.

[CohenB] LEARNING CONCEPT DESCRIPTIONS AND PROGRAMS FROM EXAMPLES

Personnel: Brian Cohen, Claude Sammut. Department of Computer Science, University of New South Wales, P.O. Box 1, Kensington, Australia, 2033. In association with Ranan Banerji, 7612 Woodlawn Avenue, Melrose Park, PA 19126.

Objectives, methods, and results:

Learning concept descriptions from examples or under the guidance of a trainer. The concepts may represent programs which can be executed to produce some output.

Concepts are represented as statements in a form of predicate calculus including existential quantifiers. Example objects (described in terms of property lists) are presented to the system. From these examples, the program produces a trial concept i.e., a boolean expression which is true if and only if a given object satisfies the predicates in the expression.

When under the guidance of a trainer, the program attempts to generalize the trial concept. It then constructs an object satisfying the new concept and shows this to the trainer. He is asked if the object belongs to the concept to be learnt. If the object does belong to the concept, then the generalization is correct, otherwise a new attempt must be made. In this way, a trainer need never know the internal representation of the concept. As well, by constructing its own examples, the program can direct the learning onto what it considers the most fruitful path.

A large part of the system has already been implemented in PROLOG.

References:

1. Banerji, R. Using a Descriptive Language as a Programming Language, *Fourth International Joint Conference on Pattern Recognition*, 1978.

2. Cohen, B. Program Synthesis Through Concept Learning, *International Workshop on Program Construction*, 1980.

3. Cohen, B. and Sammut, C. Pattern Recognition and Learning with a Structural Description Language, *Fourth International Joint Conference on Pattern Recognition*, 1978.

4. Sammut, C. and Cohen, B. A Language for Describing Concepts as Programs, *Language Design and Programming Methodology*, Springer-Verlag Lecture Notes in Computer Science, Vol. 79, 1980, J. M. Tobias (Ed).

[CohenD] KNOWLEDGE BASED THEOREM PROVING AND LEARNING

Personnel: Donald Cohen

Objectives and methods:

The topic of the research is learning as applied to theorem proving. The main objective was to obtain the advantages of knowledge based systems without sacrificing generality. The general approach is to have the system accept new information (mostly in the form of theorems about a domain that was not pre-programmed) and try to "build it in" to procedures that might have been written by a programmer to embody the given information. The underlying representation of knowledge into which the theorems are translated features reductions (simplifications), problem solving methods (which typically backward chain to generate subgoals) and demons (which reason forward from known results or hypotheses to new conclusions).

Results:

One particularly pleasing result is a domain independent program for translating theorems into demons. This involves analysis to determine which demons (if any) should be created and the order in which they should be run. The other major problem is the order in which to try problem solving methods. The results here were adequate for the problems that were tried but are not as pleasing (to me, at least) as those dealing with forward reasoning. Probably the main contribution here is the development of a sensible interpretation of best-first search in the context of theorem proving (or other problem solving that is more complex than traditional best-first search domains). Other topics include interesting new ways to use a forward reasoning facility, communication between different parts of a proof attempt and searching for an object that satisfies two different properties.

Support:

ARPA (The work was done as a graduate student at CMU.)

References:

Knowledge Based Theorem Proving and Learning , 1980 (PhD Thesis) Donald Cohen, Carnegie-Mellon University, Computer Science Dept. Technical report CMU-CS-80-115

[DeJong] TASK-INDEPENDENT LEARNING TECHNIQUES

Personnel: Kenneth DeJong (director), Steve Smith, and Art Wetzel. Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, 15260.

Objectives, methods, and results:

An inter-disciplinary research group in machine learning has been formed over the last several years in the Computer Science Department at the University of Pittsburgh. The primary goal of the group is to explore and develop task-independent learning techniques. The major focus of attention has been a set of techniques based on the adaptive system studies of Holland [2]. Two sets of results have been obtained to date. First, these techniques have been analyzed in the relatively abstract domain of function optimization. That is, the task at hand is to rapidly

accumulate knowledge about an unknown search space in order to quickly find "good" points. These results are summarized from a "control theory" point of view in [1]. Second, these techniques have been applied to the problem of generating production system programs which perform a given task. The results of their application to a Holland-Reitman maze task and a Waterman poker player task are described in [3].

Current work involves the application of these techniques to the problem of designing adaptive heuristics to NP-hard problems. By adaptive we mean techniques which are capable of accumulating knowledge about any particular instance of an NP-hard problem in such a way as to rapidly produce good approximate solutions.

References:

1. DeJong, Kenneth, "Adaptive System Design - A Genetic Approach", IEEE SMC, September, 1980.

2. Holland, John, *Adaptation in Natural and Artificial Systems* University of Michigan Press, 1975.

3. Smith, Steve, "A Learning System Based on Genetic Algorithms", Ph.D. Thesis, Department of Computer Science, University of Pittsburgh, 1980.

[Findler] TOWARD AUTOMATIC ANALYSIS AND SYNTHESIS OF STRATEGIES

Personnel: Nicholas V. Findler. Department of Computer Science, State University of New York at Buffalo.

Objectives, methods, and results:

Over the past few years, a fairly large group of students and myself have conducted research on decision-making under uncertainty and risk, problem solving and learning processes. These studies have also covered how deductive and inductive inferences are made, how heuristic rules are formed and optimized, and strategies are taught, learned and improved. While we have primarily emphasized machine cognition and machine intelligence, we have also studied and analyzed human behavior, mostly to obtain insights into the subtleties of competitive strategy and ideas for new models of competitive behavior.

The vehicle of these investigations has been the game of Poker, which has been widely studied also by mathematicians, economists and psychologists. Our models of judgment, choice and decision-making are incorporated in a complex programming system. Components of this system represent both descriptive and normative theories of behavior. We have also established a fairly flexible man-machine environment in which a total of eight human and machine players can compete with each other. The strategies of the different player programs (over 30 at present) respond to game situations according to different ordered sets of decision premises. Some of these have benefited from our findings about how human players rely on various heuristics and ill-defined decision criteria. Others are pure "machine intelligence"-oriented in that they act in accordance with mathematical and logical rules that are not necessarily followed, at least explicitly, by human players. The underlying reasoning processes range from relatively simple conceptual mechanisms to very complex constructs. A fairly detailed, but somewhat dated description of some of the static and learning strategies available in our system can be found in [9, 11].

In addition to some work in system development, there are three areas under active study:

1. The **Quasi-Optimizer** which will study the behavior of competing strategies, infer their internal mechanisms, and construct a descriptive theory of each. It will then evaluate the effectiveness of the components of these model strategies and generate a master strategy -- a normative theory -- from the most effective components.

2. The **Advice Taker/Inquirer** will be able to generate the algorithm of a competitive strategy from the advice given by a human instructor in terms of principles and high-level examples. The system would make inquiries whenever the advice is vague, incomplete or contradictory.

3. The **Pattern Recognition Strategy** will establish rules of pattern formation for given sequence of events (morphs), inductively infer the values of variables that are "hidden" via stochastic relations gradually constructed between causally connected observable and non-observable variables, and exploiting this knowledge in a strategy. The stochastic relations are in the form of generalized production rules.

References:

Findler, N. V., H. Klein, W. Gould, A. Kowal, and J. Men "Studies on decision making using the game of Poker," *Proc. Congress 71*, Amsterdam: North-Holland, pp. 1448-1459, 1972.

Findler, N. V., H. Klein, and Z. Levine, "Experiments inductive discovery processes leading to heuristics in a P program," in *Cognitive Processes and Systems*, M. Beckm G. Goos, and H. P. Kunzi, Eds., Berlin: Springer, pp. 257- 1973.

Findler, N. V., "Computer experiments on forming optimizing heuristic rules," in *Artificial and Human Thin* A. Elithorn and D. Jones, Eds., Amsterdam: Elsevier, pp. 177- 1973.

Findler, N. V., H. Klein, R. C. Johnson, A. Kowal, Z. Le and J. Menig, "Heuristic programmers and their gam machines," *Proc. ACM National Conf.*, San Diego, CA, pp. 2 1974.

Findler, N. V., "Design of an interactive environment to s the behavior of several robots which can learn, plan their ac and co-exist," in *Computer Oriented Learning Processes*, J. C. Sir Ed., Groningen: Noordhoff, 1974.

Findler, N. V., "Studies in machine cognition using the of Poker," *Comm. ACM*, vol. 20, pp. 230-245, 1977.

Findler, N. V., "Computer Poker," *Scientific American*, 144-151, 1978.

Findler, N. V. and J. van Leeuwen, "The complexit decision trees, the Quasi-Optimizer, and the power of heu rules," *Information and Control*, vol. 40, pp. 1-19, 1979.

Findler, N. V., G. Sicherman, and S. Feuerstein, "Tea strategies to an Advice Taker/Inquirer system", *Proc. Euroli Conf.*, London, England, pp. 457-465, 1979.

Findler, N. V., "Pattern recognition and gener production systems in strategy formation", *Proc. Fifth Internat. on Pattern Recognition*, Florida, FL, pp. 140-145, Vol. I, 1980.

Findler, N. V. and J. P. Martins, "On automating com

model construction, the second step toward a Quasi-Optimizer system", accepted by the *Journal of Inform. and Optimization Sciences.*

Findler, N. V. and E. Morgado, "Optimum decision rules for multiple sequential trend analysis", in preparation.

[Freuder] LEARNING FROM EXAMPLES

Personnel: Eugene C. Freuder, University of New Hampshire.

Objectives, methods, and results:

I am interested in the general area of learning from examples, specifically extending the Winston model [1].

I. Variations on induction [2].

1. "Rash induction", e.g., assuming that any shape block will do for the top of an arch, from the first example, and later specializing, if necessary; as opposed to assuming the top must be e.g. a brick, and later generalizing.
2. "Deductive induction", e.g. assuming that all grungi fruits are purple after seeing a single purple grungi fruit, using the rule: "all fruits with the same name have the same color". (Of course this is wrong; it's still a good rule.)

II. Machine generation of examples.

The aim is to either reduce the role of the teacher, or to assume that role. Generating examples, then asking for a response from the teacher, can finesse the question of potential matching problems [3].

Issues:

1. Composition and order. Choosing a *minimal* sequence of examples sufficient to identify the concept.

2. Relevance. Making explicit the properties and relationships to be used in a given problem area, e.g. Winston's world requires determination of support relationships, but not color properties. Utilizing constraints among properties and relationships to reduce the number of examples required, e.g. if a "support" relationship between two objects is required, the "above" relationship is redundant or implied, and does not need to be investigated.

3. Backup. Eliminate backup by carrying alternatives along. E.g. when we see arches with bricks and wedges for top pieces, we do not have to make a specific choice of how to generalize; we do not have to choose between must-be-standing and must-not-be-lying when we see a lying brick near miss. Brick and wedge (standing and lying) are members of a set of possibilities; we gradually rule out possibilities. "Must-be" simply means that we have reduced the set to a single possibility. This approach need not require an impractical storage burden. For example, [4] indicates that for lattices of possibilities, two points, an upper and lower bound, are sufficient to identify the possibilities that have not been eliminated (see also [5]).

III. Simple formal model.

The following simple model may be useful for exploring some of the issues raised above. A concept will be comprised of a set of mutually exclusive atomic elements, e.g. for "color" we have {red, blue,...}. A concept family will be defined as a subset of the power set of a concept. For example, the concept family for color includes "warm color" = {red, orange,...}, "primary color" = {red, blue,...}, etc. Now the example generating problem for a concept S in the family of C becomes one of choosing the smallest set {c(i)} of atomic elements of C such that answers to the questions "is c(i) S?" will uniquely determine S. The contents of the family can depend on the problem context. For example, if warm color and primary color were the only concepts in the family of color relevant to some learning domain, the question "orange S?" would determine which concept was being exhibited.

References:

1. Winston, P. H. Learning structural descriptions from examples, in *The Psychology of Computer Vision*, P.H. Winston, ed. McGraw Hill, New York. 1975.

2. Freuder, E. C. Hypothesis guided induction: jumping to conclusions. *Proc. of the Second National Conference of the Canadian Society for Computational Studies of Intelligence.* 1978.

3. Knapman, J. A critical review of Winston's Learning Structural Descriptions from Examples. *AISB.* October 1978.

4. Young, R. M. and Plotkin, G. D. Analysis of an extended concept- learning task. *Proc. of the Fifth International Joint Conference on Artificial Intelligence.* 1977.

5. Brayer, T. M. and Fu, K. S. Some multidimensional grammar inference methods, in *Pattern Recognition and Artificial Intelligence*, C. H. Chen, ed. Academic Press, New York. 1976.

[Haas] KNOWLEDGE ACQUISITION RESEARCH AT SRI

Personnel: Norman Haas and Gary G. Hendrix, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025. Other major contributors to the KLAUS project are or have been: Robert Moore, Mabry Tyson, Beth Levin, Stan Rosenschein, Barbara Grosz, Jane Robinson, Pat Hayes, and Jerry Hobbs.

Objectives, methods, and results:

At SRI, we are currently conducting research on how to enable computer systems to acquire large bodies of facts about totally new domains from domain experts - "tutors" - who have little or no training in computer science [1],[3],[4],[5]. We have implemented a pilot system, and identified several research issues, discussed below.

We are most concerned with acquiring information needed to support "question-answering" or "fact-retrieval" activities. In particular, our interest is in collecting and organizing individual facts about new domains, rather than rules for judgmental reasoning. The method of acquisition we are using is "learning by being told," as opposed to "learning by example."

Our goal is to create computer-based Knowledge Learning And Using Systems (KLAUS systems) that can aid their users in managing information. A KLAUS should be able to talk to a user about his problems and subsequently apply other types of software, such as data base management systems, report generators, simulators, etc, to meet his needs. We think interactive dialogs in natural language are the most convenient means for obtaining most of the application-specific knowledge needed by such systems.

To create such KLAUSes, we are addressing several fundamental research problems:

1. A powerful natural-language processing capability is

required. In particular, it must operate in the context of an incomplete knowledge base.

2. Seed concepts and seed vocabulary, with which tutors can describe the concepts of new domains, must be identified for inclusion in the core system.

3. A structure for lexical entries must be specified so that the system can acquire new lexical information. This is a very challenging task for certain categories of words, particularly verbs.

4. The linguistic constructions that people use in introducing new concepts, ranging from simple syntactic patterns to complex uses of analogy, must be identified and analyzed so they can be interpreted correctly.

5. A flexible scheme of knowledge representation having general expressive power is necessary. It should include inherent features that can aid in organizing knowledge and in supporting the incremental acquisition of knowledge.

6. An efficient, general problem-solving capability is needed to answer questions and to draw inferences for integrating newly acquired information. (Acquiring domain-specific problem-solving procedures is a separate research question.)

7. Procedures for integrating new concepts into the system's knowledge base must be designed. Because tutors will often provide only partial descriptions of new concepts, methods have to be devised for deciding what additional facts must be obtained from the tutor to insure proper linkage between the new concepts and those previously acquired.

8. A set of readily understandable questions is needed for eliciting information from tutors. The length and number of questions should be minimized to impose as small a burden on tutors as possible.

We are trying to integrate our solutions to these problems, balancing the requirements of one facet of the system against those of others.

We have recently developed a pilot KLAUS, called NANOKLAUS. Its principal components are a natural-language processing module based on the pragmatic grammar LIFER [2], a formal deduction module that operates on a data base of well-formed formulas (wffs) in a many-sorted first-order logic, and a set of specialized support procedures that aid in acquiring domain and lexical knowledge.

Future Plans:

At this time NANOKLAUS is best described as a fragile, proof-of-concept system still in its early development. In the coming year, we plan to greatly extend its linguistic coverage by replacing LIFER with Robinson's DIAGRAM grammar [5]. Once this has been accomplished and NANOKLAUS's verb acquisition package extended to accept particles and prepositional phrases, we believe the system will be able to serve as a useful tool for aiding AI researchers in the construction of knowledge bases for other AI systems.

We plan in the near future to enable NANOKLAUS to access a conventional data base management system. In this configuration, a user should be able to tell NANOKLAUS about a new domain, about a data base containing information pertaining to that domain, and about the interrelationship of the two. The new system would then be able to use the data base in answering questions regarding the domain.

Farther into the future, we hope to extend NANOKLAUS capabilities to include learning by analogy, acquiring and reasonir about the internal structures of processes, dealing with causalit and dealing with mass terms.

Support:

Preparation of this paper was supported by the Defen Advance Research Projects Agency under contr N00039-79-C-0118 with the Naval Electronic Systems Commar The views and conclusions contained in this document are those the authors and should not be interpreted as necessar representing the official policies, either expressed or implied, of t Defense Advanced Research Projects Agency of the United Sta Government.

References:

1. N. Haas and G. G. Hendrix, "An Approach to Apply and Acquiring Knowledge", Proceedings 1st National Conference Artificial Intelligence, Stanford, Cal, (August 1980).

2. G. G. Hendrix, "The LIFER Manual: A Guide to Buil Practical Natural Language Interfaces," Technical Note Artificial Intelligence Center, Stanford Research Institute, M Park, California (February 1977).

3. G. G. Hendrix, "Encoding Knowledge in Partitic Networks," in "Associative Networks - The Representation and of Knowledge in Computers, N. V. Findler, ed. (Academic P New York, New York 1979).

4. R. Moore, "Reasoning from Incomplete Knowledge Procedural Deduction System," MIT Artificial Intellig Laboratory, AI-TR- 347, Massachusetts Institute of Techno Cambridge, Massachusetts (1975).

5. J. J. Robinson, "DIAGRAM: an Extendible Gramm Natural Language Dialogue," Technical Note 205, Art Intelligence Center, SRI International, Menlo Park, Calif (February 1980).

[Harris] TECHNIQUES UNDERLYING AN ARCHITECTURAL EPISTEMOLOGY FOR CONSTRUCTING APPROPRIATE NEW MEANIN

Personnel: Gregory Harris (Greg.Harris at CMU-Computer Science Department, Carnegie-Mellon Univ Pittsburgh, PA, 15213, USA

Objectives and Methods:

I am interested in the meanings of symbols and how s gain their meaning. Assuming cognitive problem solving act implemented as symbol processing, I am unwilling to processing to be heedless of the meaning of what it is doing symbols. It strikes me that approximating a thinker with a cruncher is not a good approximation if ludicrous and contr garbage causes no trouble in the engine. The approximatior improved by having the system halt and summon its program

Within the cognitive architecture, then, there ough provision and cause for handling each symbol with and through its connection to whatever gives it "meaning". If a names a search that may be done, the meaning is what t performing that search. If a symbol names a real externa the meaning is how the system connects up the symbol entity with symbols for the rest of what it knows. In this changeability of what to think when the processor runs acr

symbol (loosely, a kind of private meaning for the symbol) can become the subject matter for a processing model, such as we find in AI at present for ordinary external topics. I call this approach to flexible symbols "architectural epistemology" because the computer, in terms of which the symbol processing is specified and built up and controlled, adheres to a designed-in goal: that of keeping the cognitive function (at least the relevance) of the meanings of currently important symbols also in view. The subject matter of my work on learning is how to produce an epistemology for a family of problems. Expertise in an epistemological domain would mean that becoming more familiar with the family of problems (acquiring the useful facts, skills, subgoals, and interactions that make up the reusable cognitive context of capable problem solving) would happen through gaining experience with the symbols and their meanings. One definition of "cognitive function" is provided by the need to control attention in the problem solver. I am trying out the view that attending to something in an environment means working with a model of it.

The methods I am presently entertaining are: to use a network of constraints to simulate the low-level operation of various meanings; to use contradictory or otherwise troublesome simulation states as a cause for constructing new symbols; to use symbolic expressions as the names of typed compositions of the modelling methods named by the subexpressions; to use expression compositions with a built-in epistemological commitment; to use as models mechanisms that augment the network of constraints that represents what is being simulated. The "learning" is the convergence of these model-naming expressions and these model-running structures on one another. I've been writing these up, albeit slowly and with much revision. There are other ways to put this together and get learning. For instance, "learning" might happen when a new model is abstracted over portions of the network that regularly converge on the same meaning, if somehow the rediscovery of subproblems along the way gets replaced by the usual solution to them. Efficient simulation of rediscoveries lies at the heart of the idea of naming compound models to work with. Covering the internal environment with an epistemology, as for an external environment, offers a way of picking up new models.

Results:

I have made a brief contribution to this issue because my approach to learning is different from others I know of, although there are no experimental results in learning by computer I can report yet. This would be a huge system, although the manageability of it cannot be estimated from the labor that goes into understanding examples of learning to solve unfamiliar problems in terms of an unknown but fully-operational architectural epistemology. The latter needs to be automated, and I would like to do so.

Future Plans:

I intend to remain interested in this topic, as well as in most simpler versions of it. Related explicit control problems exist when simultaneously building and describing a labeled directed graph, for suitably interesting uses of graphs. I believe the controlled circumscription of graph structure would be a usable stepping-stone for me.

Support:

I am a graduate student in AI at CMU CSD working on my own. As this issue went to press, I was on leave to work on CMU's personal computing environment project, another interest of mine.

References:

No references to me, but see pointers into the literatures on problem solving, epistemology, control of inference, explicit control, constraint systems, truth maintenance, internal environments. I make no attempt to cover the vital topic of finding the right abstractions over composite structures per se. I believe that is mostly an open subject, yielding only to sharp formulations.

A. Newell and H.A. Simon, Human Problem Solving (Prentice-Hall, 1972).

R. Brachman and B.C. Smith, eds. SIGART February 1980 special issue on a survey about Knowledge Representation.

D. Bobrow, ed. special issue on nonmonotonic logics, Artificial Intelligence, vol 13, nr 1 & 2, April 1980.

D.G. Bobrow and D.A. Norman, "Some Principles of Memory Schemata," in Bobrow and Collins, eds., Representation and Understanding (Academic Press, 1975).

R. Davis, "Meta-Rules: Reasoning about Control", Artificial Intelligence, vol 15, nr 3, December 1980.

J. de Kleer, et al. "AMORD: Explicit Control of Reasoning", SIGPLAN/SIGART Proceedings of the Symposium on AI and PL, 1977.

D. McAllester, "Outlook on Truth Maintenance", MIT AI TR 551, 1980.

J. Doyle, "A model for Deliberation, Action, and Introspection", chapters 3 and 5, MIT AI TR 581, 1980.

G. Steele, "The Definition and Implementation of a Computer Programming Language based on Constraints", MIT AI TR 595, 1980.

H.A. Simon, "The Architecture of Complexity", in The Sciences of the Artificial, (MIT Press, 1969).

[Harrison] LEARNING CONTROL OF PRODUCTION SYSTEMS

Personnel: Malcolm C. Harrison, New York University. Salvatore J. Stolfo, Columbia University.

Objectives, methods, and results:

The aim of our research is to develop a problem-solving methodology and system capable of learning heuristic control information by observing the performance of a human expert. The system contains the following components: a Production System (PS) language in which the knowledge of the problem domain is encoded by a human expert (in terms of a long term production memory, a short term memory of current assertions, and an interpreter executing the standard recognize/act cycle), a control language facility stored in its own knowledge base and which communicates with the PS interpreter controlling the selection of relevant productions during execution, and inductive inference procedures which construct procedures in the control language. By running the system in training mode, a human expert provides sample problems and solution sequences making all decisions (selection of productions) during execution of the problem-solving program. The observed traces are then analyzed by the inference procedures producing heuristics (control strategies, plans) for use by the system alone for subsequent problems.

In the initial system we implemented, we defined a control language, CRAPS, based on a Regular Expression formalism. Using regular expressions is obviously restrictive (i.e. no self-embedding procedures) yet it is powerful enough in general to describe interesting heuristics. As an example, the set of context-free string productions: P1: S-->ABC, P2: A-->aA, P3: B-->bB, P4: C-->cC, P5: A-->a, P6: B-->b, P7: C-->c, where {A,B,C,S} are nonterminals and {a,b,c} are terminals, when used with the initial sentential form S, generates the context-free language {a^i b^j c^k | i,j,k>=1}. If we restrict the permissible sequences of rule applications to be a member of the language generated by the regular expression P1 (P2 P3 P4) P5 P6 P7, the resulting language generated by the above grammar is {a^n b^n c^n | n>=1} which is context-sensitive.

The CRAPS language has additional facilities specifying conditions under which con trol operations should be applied (eg. when a repetition should stop) which resulted in a language difficult to classify in terms of the Chomsky hierarchy.

Using regular expressions, the inductive inference problem can be stated as the problem of constructing the minimum length regular expression consistent with the sample inputs. Since the system was trained with an incomplete set of solution sequences as examples, the resulting problem is NP-complete; a well known result. Our inference procedures were necessarily heuristic in nature (sample traces were longer than 400 symbols), based on data flow analysis and divide-and-conquer search. See [2] for complete details.

The initial system was applied to two toy problems with good success: a binary tree traversal program and a jigsaw puzzle-solving program. The preliminary experiments uncovered several flaws with the language. In particular its rigid form of control (used in an irrevocable way with no backtracking) lacked robustness, not allowing for alternative courses of action when appropriate. For that reason we included an additional debugging facility represented as a set of 'meta-rules', which allowed for a more flexible control regime and resulted in completely new problems being solved by the system. The inference procedures were augmented to include procedures which inferred these meta-rules from fragments of the original sample traces.

We are currently experimenting with a new control language MCL, which does not have as rigid a structure as CRAPS and which allows for an easily modified and extensible knowledge base for the control information. An MCL program is represented as a set of meta-rules which are relation revision productions. The relations or predicates we have defined represent the state and actions of the problem-solving system. During training, the human expert is allowed to view only the output of any fired productions, the current set of active rules and the previously fired rule, and make the appropriate selection of rules from this information alone. The predicates we have defined model these aspects of the system. Consequently, each action by the trainer provides a 'raw' meta-rule. This set of raw meta-rules are then the objects for the subsequent inference procedures.

Presently, we have implemented procedures to cluster the raw meta-rules for generalization purposes. (The resulting generalized rules comprise the MCL program.) The generalization process is based on Vere's [5] procedure for inducing relational productions with an additional facility to undo incorrect (over-)generalizations, similar to the self-correcting system described by Whitehill [6]. If the MCL control program fails (either because a meta-rule is active that should not be, or one is not active that should be) the stored raw meta-rules used to construct the failing rule would be regeneralized using the current state of the system as a (negative or positive, as the case may be) example. With this approach, the acquisition and learning of control knowledge (plans, heuristics, strategies, etc.) is viewed as an ongoing evolutionary process durir the life time of the system, guided by a human expert and capab of correcting itself when necessary.

We are currently applying our system to rob problem-solving and VLSI design problems. The initial grou work is in progress to build and debug the PS knowledge bas One particular application in VLSI design we are considering is t analysis of circuit specifications (in the Caltech Intermediate Fo language) to identify functional components and possible flaws the specifications of those components. Additionally, we experimenting with a system for graduate student advisement Computer Science.

Support: Office of Naval Research Contr N00014-75-C-0571.

References:

[1] Dietterich, T. and Michalski, R., Learning Generalization of Characteristic Descriptions: Evaluation Crit and Comparative Review of Selected Methods, Proc. IJCAI Tokyo, Japan, 1979.

[2] Stolfo S., Automatic Discovery of Heuristics Nondeterministic Programs from Sample Execution Traces, Pl thesis, Courant Institute Computer Science Report #18, Oct 1979.

[3] Stolfo S. and Harrison M., Automatic Discovery Heuristics for Nondeterministic Programs, Proc. IJCAI 6, To Japan, 1979.

[4] Stolfo S., Learning Control of Production Syst Columbia University Computer Science Report, Computer Sci Dept., 1980, (submitted for publication).

[5] Vere S., Induction of Relational Productions in Presence of Background Information, Proc. IJCAI 5, MIT, 1977

[6] Whitehill S., Self-correcting Generalization, Proc. AA Stanford, 1980.

[Hayes-Roth] LEARNING THROUGH KNOWLEDGE PROGRAMMING AND KNOWLEDGE REFINEMENT

Personnel: Frederick Hayes-Roth, Philip Klahr, and J. Mostow. The Rand Corporation, 1700 Main Street, Monica, Ca. 90406.

Objectives and Methods:

We have come to view learning chiefly as a probl modeling, planning, and debugging. Most knowledge we h acquired from a combination of instruction and practice. We that when an instructor provides advice to a learner, the must develop a model to interpret that advice and the behavior that can carry that advice out. Even when the ad imperative and operational from the instructor's viewpoint, "Don't drop crumbs" or "Win every point," the learner often solve a difficult problem to operationalize the advice. One for the difference in perceived difficulty in carrying out ad that the instruction is based on the instructor's domain mod the learner frequently has a different model or no model domain at all.

As a consequence of the disparity between models, the

must construct domain models and attempt to reason about achieving effects in the domain from this domain model. A variety of errors can arise when the learner executes its planned actions. Undesired results may reflect a deficiency in the domain model or in the instrumental reasoning behind the planned actions.

The term knowledge programming refers to the process of converting advice into procedures. Since these procedures may have bugs, the learner must rectify them through a process we refer to as knowledge refinement.

When the learner is faced with evidence of error, it needs to diagnose and repair the faulty knowledge. This knowledge may be diagnosed using heuristics that apportion blame to knowledge units supporting refuted expectations. This means maintaining dependencies between knowledge units that underlie planned acts and backtracking from learning situations to those whose expectations are violated but whose premises are not. These units of knowledge are faulty.

When a unit of knowledge is diagnosed as faulty, it must be modified. A variety of heuristic methods have been developed for this kind of knowledge refinement. If every knowledge unit is viewed as a rule with prerequisites (premises) and entailments (expectations), to repair a faulty rule we must change the premises or the expectations. The premises can be made more restrictive to exclude the rule's applicability from the error-producing situation. Or, the expectations can be made more general to assimilate the specific outcome that occurred but which disconfirmed the original expectation.

The learning process is an iterative cycle that repeats once the learner has conjectured new knowledge elements. These elements become part of its "knowledge base" or model. A revised model can support new interpretations and operationalizations of old advice. Old plans may depend on aspects which no longer pertain, or opportunities may arise for improved plans in light of the new knowledge. Thus, the learner may "advise" itself, and this in turn reinitiates knowledge programming and knowledge refinement.

Our method for developing these ideas is to construct experimental systems for knowledge programming and knowledge refinement.

Results:

Mostow has implemented for his dissertation a semi-automated operationalization system under Hayes-Roth's supervision. This system converts imperative advice or constraints into operational, effective procedures. Hayes-Roth and Klahr have implemented a system for the card game hearts to test diagnostic and knowledge refinement heuristic rules. Hayes-Roth has implemented a general dependency maintenance system for recording plan rationales and monitoring effects of changed beliefs. He has also formulated several heuristics for refining units of instrumental knowledge in accordance with the ideas in the preceding overview. These heuristics use a method of proofs and refutations which is a generalization and extension of some ideas due to Lakatos.

Future plans:

This learning work is a constant but low-level effort at Rand. Knowledge acquisition, knowledge programming, and knowledge refinement is what we AI professionals due for a living. We pursue these goals of automated aids for this work continually, but currently not in any separated or independent project context.

Support:

Our previous work has been supported by a grant from the Intelligent Systems program at NSF, by Rand's Project Air Force, and from the Information Processing Techniques Office of DARPA. We have several current sources of support for this line of work. AI researchers who may be looking for opportunities to pursue related ideas should contact F. Hayes-Roth at Rand Corporation.

References:

Hayes-Roth, F. Theory-driven learning: proofs and refutations as a basis for concept discovery. N-1543, The Rand Corporation, Santa Monica, CA, June 1980.

Hayes-Roth, F. Probabilistic dependencies in a system for truth maintenance and belief revision. WD-947-ARPA, The Rand Corporation, Santa Monica, CA, February 1981. Submitted to IJCAI-81.

Hayes-Roth, F., Klahr, P., and Mostow, D.J. Knowledge acquisition, knowledge programming, and knowledge refinement. R-2540-NSF, The Rand Corporation, Santa Monica, CA, November 1979.

Lakatos, I. Proofs and refutations: The logic of mathematical discovery. Cambridge University Press, Cambridge, 1976.

Hayes-Roth, F., Klahr, P., and Mostow, D. J. Advice-taking and knowledge refinement: an iterative view Hayes-Roth, F., Klahr, P., and Mostow, D. J. Advice-taking and knowledge refinement: an iterative view of skill acquisition. Sixteenth Annual Carnegie Symposium on Cognition. Pittsburgh, PA, 1980. (Also to appear in, Learning and Cognition, J. R. Anderson (Ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.)

Mostow, D. J. , & Hayes-Roth, F. Machine-aided heuristic programming: A paradigm for knowledge engineering. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan, 1979. Also N-1007-NSF, The Rand Corporation, Santa Monica, CA,

Mostow, D. J. Mechanical transformation of task heuristics into operational procedures, Ph.D. Thesis, Carnegie-Mellon University, 1981, in progress.

[Herman] AUTOMATIC LEARNING OF DESCRIPTIONS OF 3-D OBJECTS

Personnel: Marty Herman, Computer Science Dept., Carnegie-Mellon University.

Objectives and Methods:

The ultimate goal of this research is to develop a system that learns models of 3-D objects. For example, we would want it to learn the concept model "chair" and use it to recognize chairs it has not seen before. The learning approach adopted is "learning from examples." However, we realize that the system must have a good 3-D description of each sample before model learning can occur.

Our first objective is therefore automatic learning of the 3-D shape of an object. Our approach is to represent objects in terms of surfaces in Kanade's "origami" world, and generate partial 3-D shape descriptions from each of several views of the object. The problem then becomes one of integrating all the partial descriptions so as to arrive at a unique and complete 3-D description.

Our method involves hypothesizing hidden surfaces for some of the partial descriptions, generating projections of these descriptions along the directions of other views, and then seeing how well the projections match the descriptions derived from the latter views. The best matching partial descriptions together form the complete 3-D description.

Future plans:

The second objective of this research is automatic learning of 3-D concept models from examples. We intend to use an interactive approach, wherein the system can generate its own samples and receive classification feedback from the teacher. In this manner, the system can hypothesize certain properties and relations as being relevant, generate samples to test these hypothesizes, and update its current model based on the response of the teacher.

Support:

Office of Naval Research

References:

Herman, M. Towards learning descriptions of three-dimensional objects. Papers of the Workshop on Current Developments in Machine Learning, Carnegie-Mellon University, Pittsburgh, PA, July 1980.

[Holland] THEORY AND MODELLING OF ADAPTIVE PROCESSES IN COGNITION AND KNOWLEDGE ACQUISITION

Personnel: Professors John H. Holland, Judith S. Reitman, and Paul Scott; Dr. Kenneth A. Winter; 3 Doctoral Candidates. (Logic of Computers Group, Dept. of Computer and Communication Sciences, The University of Michigan. Chicago-Michigan Center for Cognitive Science).

Objectives and Methods:

Our research in this area centers on adaptive processes as learning algorithms for knowledge acquisition and cognitive systems. The project is concentrating on the following problems in cognition and learning: (1) development of appropriate decision and action *sequences* when payoff or reinforcement occurs only after long sequences of correct context-dependent actions; (2) acquisition and use of a cognitive map based on system experience with the task domain; (3) transfer of learned generalizations and associations from one task to another.

The systems being studied are oriented around the processing of messages. The interface with the system's environment (task domain) produces messages that serve as input. The performance part of the system consists of a set of productions. The action of any production is a message that is a function of the messages satisfying its conditions. The system's outputs (effectors) are controlled by messages. Any number of productions can be active simultaneously so that any number of messages can be processed (and produced) at a given time. The state of the system's performance part is given thus by the current list of messages.

Learning for these systems is mediated by two kinds of algorithms. The first kind of algorithm, the *bucket brigade algorithm*, assigns weights to productions. The weight assigned to a given production is tied, by the algorithm, to that production's cumulative value as a part of the system's mechanism for generating action sequences. This weight determines the likelihood that the production will be activated when its conditions are satisfied. The second kind of algorithm, the *genetic algorithm*, is used to generat new productions for test on the basis of system experience. Th weights assigned by the bucket brigade algorithm serve to bias th application of the genetic algorithm.

It can be proved that the genetic algorithm favors th construction of new productions from "building blocks" (schemat that appear as components in the definition of productions that ha already proved useful to the system (see reference [1]). Furth theoretical work makes it clear that newly generated productions c be introduced without seriously disturbing capabilities alrea well-established. There is also a strong indication from t theoretical work (though this must be demonstrated in t simulations) that the system will automatically organize itself in default hierarchy. As a result, the actions taken at any time shou depend upon the specificity of the information available to t system (as measured by the specificity of the conditions defined the current set of productions.)

Results:

There exists a considerable body of theory underpinning use of genetic algorithms (see references [1] and [2]). We h also completed a series of tests using a simulation based o simplified version of the planned organization (reported in refere [3]). The environments in this simulation were mazes involving to a dozen independent choices. Each choice point characterized by a unique bit string (the input message). system employed 100 single condition productions. The object to test the system's ability to uncover and use regularities in information received as the system moved about in the m (E.g., in the input bit string, a 1 in position 2 coupled with a position 4 might consistently signal that a "left turn" shoul made). The results were positive showing (1) learning an orde magnitude faster than with weight-changing techniques alone, (2) transfer of learning from maze to maze.

Future Plans:

We are currently finishing a "test-bed" simulation that pe (1) up to 1000 2-condition productions, any 32 of which ca active on a given time-step, (2) an environment consisting of 256 objects (each with an arbitrary set of properties) moving o 65,000 by 65,000 grid divided into 256 regions (each wit arbitrary set of "textural" properties), and (3) a time-step of 0.1 second (allowing the system to operate in "real time"). this test-bed we intend to explore the abilities of a cognitive s organized along the lines outlined above. We also expect to l good deal about systems, particularly production systems, h many control procedures active simultaneously.

Support:

The National Science Foundation. The Sloan Foundatio

References:

[1] Holland, John H. *Adaptation in Natural and A Systems.* The University of Michigan Press. Ann Arbor. 197

[2] Holland, John H. Adaptation. In *Progress in The Biology 4.* Rosen, R. and Snell, F. M. (eds.) Academic Pres York. 1976.

[3] Holland, John H. and Reitman, Judith S. C Systems Based on Adaptive Algorithms. In *Pattern-Inference Systems.* Waterman, D.A. and Hayes-Roth, F Academic Press. New York. 1978.

[4] Holland, John H. Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge-Bases. *Int. J. Policy Analysis and Information Systems.* 4, 2 pp. 217-40. 1980.

[Kibler] LEARNING CONTROL KNOWLEDGE FOR PROBLEM SOLVING

Personnel: Dennis Kibler, Paul Morris. UC at Irvine.
Objectives and methods:

In high school algebra only a few laws of algebraic manipulation are needed to simplify a formula or solve an equation. This task is difficult for most students although, with experience, they get better at this problem solving activity. We believe that the difficulty of this task is due to the lack of awareness of the reasons and rules for controlling the application of the laws. Teachers have no explicit form for the control knowledge and so cannot impart it to students. English is ill-suited for expressing control knowledge. One aim of our work is to create a language necessary for expressing control knowledge. To test the generality of this language a program, currently called BLOCKHEAD, will be tried on a number of different problem solving domains, beginning with the blocks world and including towers of hanoi, tic-tac-toe, simplification, and possibly code generation. It is expected that the language for expressing control knowledge will grow as various problem domains are tackled.

The measure of the program's performance will be three-fold. One measure is its ability to solve problems. Another will be the control knowledge that it generates for a particular domain. This will be measured both subjectively (how right does the control knowledge seem) and objectively (how much search does the control knowledge eliminate). Moreover the program will be expected to support its control knowledge with reasons. The reason will also be generated by the system. The reasons for its control knowledge will be evaluated subjectively.

There are a number of different forms that control knowledge can take. The following list is not meant to be exhaustive, but marks our current understanding.

DO'S: By examining a successful search, features that relate the particular solution to the goal can be inferred. These generated heuristics can be used to order the search process.

DONT'S: Heuristics of this sort tell what not to search. They are less specific then DO'S, but are usually discovered first. One way to gain them would be to generalize from loops that occur in the search space. For example, in the block's world one would soon learn that it is not helpful to move the same block in consecutive moves. Other DON'T heuristics can be abstracted from unsuccessful portions of the search.

Goal Orderings: If the objective is to achieve some conjunction of goals, the particular order that the goals are solved in may be useful in avoiding search. In the BLOCKS world a heuristic which is easily generated from examining solutions is to solve on (X,Y) before solving for on (Z,X). Once this ordering of the goals is deduced, then a simple goal regression problem solver will solve any BLOCKS world problem. More generally the system will attempt to derive rules which enable it to partially order the goals. A metarule for ordering conjunctive goals is: achieve hard goals before easy ones. The difficulty of achieving a goal can be learned through experimentation.

Action Ordering: If the reasons for making one move contain the reasons for another, then that establishes a partial ordering on moves. Through experience, other criteria for choosing one operation over another can be inferred. This represents some of the knowledge that the system gains.

Goal Liveness: At various points of a goal search space, some of the goals may become unachievable. Only live goals are given to each node so that some unnecessary computation is avoided. In a game such as tic-tac-toe, this analysis would lead to the realization that some positions are draws without having to fill the entire board with tokens.

Goal Invariance: By considering the effects of various moves on the goals, the moves can be partitioned into classes. For a tic-tac-toe board there are thousands of possible partitions. By using goal invariance the division of the board into center, corners, and sides is automatic. This partition is generated independent of either the representation of the board or of the goals. By choosing a single representative from each of the partides, a large part of the search space is avoided. While people "see" this invariance as a result of their geometric knowledge, the program can achieve the same result by using goal invariance.

Quotient Spaces: The original problem (goal, operators, and relations) can be mapped into a quotient space where some number of relations are discarded. The quotient problem will never be harder and may be simpler. A solution in the quotient space can be used as a skeleton solution for the original space. Moreover two different problems may be exactly the same in some quotient space. Consequently the quotient solution for one problem would give a skeleton solution to the other. Unfortunately a problem requiring n relations to define yields 2**n-2 possible quotient spaces, which (fortunately) form a lattice. Through experience the useful quotient spaces can be learned.

In summary, BLOCKHEAD will be given a number of problems in a particular domain. The domain and problem will be stated in a relational way, as is STRIPS or WARPLAN. The output of the system will be 1) solutions to the problems 2) control knowledge for the domain, and 3) reasons supporting the particular control knowledge generated.

Results:

Definition of several forms of control knowledge that can be obtained through experience (see above). Explicit statement of several constraints for machine learning.

Future plans:

Development of a program (coding has started) to learn control knowledge for several different domains. We hope to report on our experience in IJCAI7.

Support:

NOSC Contract.

[Klopf] ADAPTIVE NETWORK RESEARCH

Personnel: A. Harry Klopf, (Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Wright-Patterson Air Force Base, Ohio 45433); Chuck Woody, (The Center for the Health Sciences, UCLA); Nico Spinelli, Andy Barto, and Rich Sutton (Center for Systems Neuroscience Department of Computer and Information Science, University of Massachusetts at Amherst).

Objectives, methods, and results:

In conjunction with the Air Force Office of Scientific Research, the Avionics Laboratory is carrying out an intramural and extramural program in what we term "adaptive network research." The intent is to investigate alternative architectures to that of the conventional digital computer, with an emphasis on neurobiologically oriented approaches to machine intelligence.

The program differs from most artificial intelligence research in several respects: (a) the fundamental mechanisms of intelligence are sought at levels comparable to that of the single neuron rather than at psychological and linguistic levels, although the latter are not discounted, (b) natural intelligence is viewed as involving an ongoing, dynamic, temporal, closed loop relationship with the environment in contrast to the relatively static, open loop and cognitive view that characterizes many AI systems, (c) learning or knowledge acquisition is considered to be an integral part of and fundamental to machine intelligence, such that it is considered prior to or at least simultaneously with questions of knowledge organization and utilization. Regarding the relationship of this program to past adaptive network and neural network modeling research, much of the past work is considered to have been unsuccessful for reasons that are discussed in Klopf (1980) and Sutton and Barto (1980). The intramural and extramural components of the present program are summarized below.

Intramural Program - System Avionics Division, Avionics Laboratory

Harry Klopf, a principal investigator, has been developing an adaptive network theory based on a new model of the neuron. The proposed model differs from past models in suggesting that the single neuron is a goal-seeking system in its own right. This, in turn, suggests that brains should be viewed as goal-seeking systems composed of goal-seeking components. A review of most brain models reveals that brains have generally been viewed as goal-seeking systems consisting of non-goal-seeking components. The essentially open-loop view of neuronal function that has prevailed to date appears questionable as a theoretical approach. Furthermore, some of the assumptions underlying artificial intelligence research appear to have derived from an assumed open-loop view of the neuron. The results of the present research on goal-seeking networks of goal-seeking components are reported in Klopf (1980).

Extramural Program - Department of Computer and Information Science, University of Massachusetts at Amherst.

This research is being carried out by Nico Spinelli (principal investigator) along with Andy Barto, Rich Sutton, and several others at the Center for Systems Neuroscience. One objective of this research is to make difficult problems in artificial intelligence, such as image understanding and speech recognition, more tractable by deliberately endowing computers with learning capabilities suggested by recent advances in the understanding of animal learning behavior and its cellular bases. Primitive adaptive mechanisms observed in animals possess features which may well be crucial for sophisticated adaptive behavior. These features involve the fine structure of temporal contingencies, the use of expectation driven reinforcement, and the notion of local as well as global goals.

This research places a major emphasis on closing the loop between making decisions and observing the consequences of those decisions. To that end, a basic adaptive element has been developed that is capable of dealing with a wide class of environments. Promising results have been reported in Sutton and Barto (1980).

Currently, goal-seeking networks of goal-seeking elements are being constructed that are capable of: (1) system identification and control, (2) associative memory, and (3) pattern recognition ar concept formation. The results promise adaptive behavior of a kir not previously obtainable by adaptive networks. The performan obtained on these tasks will be related with that obtainable throu more standard engineering and artificial intelligence approaches.

As part of this research, investigations are also bei conducted on neural plasticity which are aimed at understandi how neurons learn (adapt) and are connected to form intellige systems (Spinelli and Jensen, 1979). The theoretical study adaptive networks is being integrated with experimental work that yielding clear cut, large, adaptive changes in natural brains achiev through simple instrumental conditioning procedures.

Extramural Program - The Center for the Health Scienc UCLA

Chuck Woody (principal investigator) is investigating adaptive mechanisms that underly single neuron function. T study of cortical neurons in the awake mammal is being favo because these cells appear most likely to be involved with adaptive mechanisms that support intelligence. Work over the p few years has indicated that adaptive changes in mammalian cort neurons are reflected in changes in their electrical excitability. T in turn, related to corresponding changes in unit activity and to development of conditioned behavior. Recent investigations h uncovered preliminary indications of a possible post-synap cholinergically induced cyclic nucleotide effect, involving more t one-third of rostal cortical neurons, via which persistent ne excitability changes can be produced. Means for assessing ne excitability, use of a cortical model system to investigate ne plasticity, and techniques for increasing the rate of adaptatior single neurons are now being studied. Recent results have b reported in Swartz and Woody (197

Future Plans:

For the future, an expanded extramural program is u consideration and the intramural program will continue with wor extend the present theory. Inquiries regarding this program ma directed to the program managers:

Lt Colonel George Irving, III
Life Sciences Directorate
Air Force Office of Scientific Research (ATTN: NL)
Bolling Air Force Base, DC 20332
Phone (202) 767-5023

Dr. A. Harry Klopf
Avionics Laboratory
Air Force Wright Aeronautical Laboratories (ATTN: AAAT)
Wright-Patterson Air Force Base, Ohio 45433
Phone (513) 255-4949

References:

Kim, H-J. and Woody, C. D. (1979) Facilitation of eye conditioning by hypothalamic stimulation, *Soc. Neurosci.* 5:319.

Klopf, A. Harry (1980) *The Hedonistic Neuron: A The Memory, Learning and Intelligence*, Hemisphere Publ Corporation, Washington, D. C., to be published, Fall, 1980.

Spinelli, D. N. and Jensen, F. (1979) Plasticity: the mir experience, *Science* 203: 75-78.

Sutton, R. S. and Barto, A. G. (1980) Toward a Modern Theory of Adaptive Net- works: Expectation and Prediction, to be published in *Psychological Review*.

Swartz, B. E. and Woody, C. D. (1979) Correlated effects of acetylcholine and cyclic guanosine monophosphate on membrane properties of mammalian neocortical neurons, *J. Neurobiol.* 10:465-488.

[Kugel] LEARNING FROM EXAMPLES: THEORY

Personnel: Peter Kugel, Computer Science Department, Boston College, Chestnut Hill, MA 02167

Objectives, methods, and results:

I am studying induction - the process by which one goes from specific observations to general ideas - from a theoretical point of view. The basic question that I ask is "What are the relationships between the information processing power of an abstract machine and its ability to generalize correctly?"

The Turing machine, which is the traditional abstract model for the digital computer, turns out to be surprisingly weak as a generalizing machine and this leads me to focus on abstract machines that are more powerful than the Turing machines. Although such machines evaluate functions (or, more properly, functionals) are traditionally thought of as uncomputable, we know, thanks to Putnam (1965) and Gold (1965), that they can be evaluated (but not computed) by machines that contain only the machinery of the computer.

I am primarily concerned with machines that can evaluate functions in sigma-1 through sigma-3 of the Arithmetic Hierarchy. Sigma-1 machines are basically our computing machines which can evaluate any (partially) computable function. We can think of them as computing machinery for which we count the *first* output that they print. They turn out to be surprisingly poor as generalizing machines. The sigma-2 machines are the trial and error machines and they can evaluate functions that sigma-1 machines cannot (e.g. they can solve the halting problem.). We can think of them as computing machinery for which we count the *last* (rather than first) output. They are much better generalizers than the sigma-1 machines and, in particular, they solve what philosophers call "the" problem of induction (or "Hume's Problem") because the generalizations they produce must be correct. (This problem can also be solved by weaker machines, called "2-trial machines".)

Sigma-3 machines, or hyper-trial-and-error-machines, are computing machinery for which we count the first output that they print infinitely often. Such machines solve what we might call "Leibnitz's Problem": "How can we know that a given machine will give sound results in this particular world?" because, unlike the sigma-2 machines, they can be "complete" (do induction properly in "all" possible worlds).

Between these levels (between sigma-1 and sigma-2 on the one hand and sigma-2 and sigma-3 on the other) there are sub-hierarchies of some interest.

I am exploring two basic ideas right now. I am trying to characterize the inductive power of various machines mathematically (an enterprise I call "mathematics") and I am studying applications of known results to clarify conceptual problems in logic, psychology and linguistics (an enterprise I call "philosophy").

I am currently focusing my mathematical attention on trying to develop a unified and simple mathematical treatment of machines more powerful than Turing machines and on more clearly characterizing the power of the machines between the sigma levels of the Arithmetic Hierarchy. I am particularly concerned with the area between sigma-2 and sigma-3 because I happen to feel that the best models for human inductive competence lie there.

My philosophical attention is currently focused on trying to give a precise account of the intuitive concept of "intelligence" in terms of abstract machines. The basic problem lies in that intelligence does not appear, at first, to be related to the usual way of looking at abstract machines. For example, the ability to play chess (which is thought to require intelligence) is within the scope of the finite automaton (which is quite weak) while the ability to multiply (which is not usually thought to require much intelligence) is beyond the capabilities of the finite automaton. My basic approach to this problem is to try to clarify the distinction between two types of intelligence due to Hebb (1949) who called them "intelligence A" and "intelligence B" and Cattell (1963) who called the "fluid" and "crystallized" intelligence.

The theoretical study of fluid intelligence (or intelligence B) suggest that it too may be two-fold. There may be an ability to learn and an ability to learn how to learn. And within this classification, intelligence may also exhibit various "dimensions". I am testing this hypothesis in a modest experiment in which I will use computers to "measure" student intelligence in the learning-to-learn dimensions and to intervene (by computer) to strengthen their abilities in those dimensions in which they are weak. This work is supported by a grant from the Arthur Andersen Fund.

The use of these various ideas to improve machine intelligence, rather than human intelligence, which is the area that is probably of greatest interest to readers of the SIGART Newsletter, seems to require a deeper understanding of the process of convergence in the limit by trial and error machines. One of the things that my work suggests is that the development of such an understanding will probably require somebody more intelligent than I am.

References:

Cattell, R.B. (1963), "Theory of Fluid and Crystallized Intelligence: A Critical Experiment", *J. Educ. Psych.*, 16.

Gold, E.M. (1965), "Limiting Recursion", *J. Symbolic Logic*, 30.

Hebb, D.O. (1949), *The Organization of Behavior*, Wiley.

Kugel, P. "A Mathematical Theory of Intelligence" (in preparation).

Kugel, P. "Beyond the Turing Machine" (in preparation).

Kugel, P. "Goedel's Theorem and Mechanism" (in preparation).

Kugel, P. (1979), "Generalizing", Boston College Working Paper 79-90 (submitted for publication).

Kugel, P. (1979), "The Controversy Goes On - Can Computers Think?", *Creative Computing*, vol. 5, issues 8, 9, 10.

Kugel, P. (1979), "Nothing Worth Learning Can be Taught", *Improv. Coll. and Univ. Teaching*, 27, 1.

Kugel, P. (1977), "Induction, Pure and Simple", *Inf. and Contr.*, 35,4.

Kugel, P. (1976), "What Can a Computer Learn From Examples?", *SIGART Newsletter*, 57.

Kugel, P. (1975), "A Theorem About Automatic Programming", *Ibid.*, 51

Putnam, H. (1965), "Trial and Error Predicates and the Solution of a Problem of Mostowski", *J. Symbolic Logic*, 20.

[Langley] REDISCOVERING CHEMISTRY WITH BACON.4

Personnel: Pat Langley, Gary L. Bradshaw, Herbert A. Simon. Department of Psychology, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

Objectives, methods, and results:

BACON.4 is a production system that discovers empirical laws. The program was not designed to replicate the historical details of the discovery process, but is intended as a sufficient model of how discovery might occur. BACON.4 is named after Sir Francis Bacon (1561-1626), the early philosopher of science. The program incorporates a small set of heuristics for finding constancies and trends in data, and for formulating hypotheses and defining theoretical terms based on these regularities. BACON.4 is intended to be a general discovery system; the data-driven, Baconian nature of its heuristics were designed with this goal in mind.

Formulating Hypotheses. Standard analyses of the scientific method partition the world into data or observations, and hypotheses or laws that explain or summarize those data. BACON.4 replaces this dichotomy with a continuum along which information is represented at varying levels of description. Thus, a description at one level acts as an hypothesis with respect to the descriptions below it, and as a datum for the description above it.

Consider some data obeying the ideal gas law. This law may be stated as $pV/nT = 8.32$, where p is the pressure on a gas, n is the number of moles, T is the temperature, and V is the volume of the gas. Suppose BACON.4 is given data showing that when p is 1, n is 1, and T is 300, the value of V is 2496.0. If the first three terms are under the system's control (independent variables), one can think of their values as conditions on the value of V (the dependent variable). Now suppose that after gathering additional data, BACON.4 finds that pV is 2496.0 whenever n is 1 and T is 300. This second level description summarizes all first level observations with similar conditions, but it can be treated as data in turn. Upon varying T, the program generates other second level summaries; these lead to the third level summary that pV/T is 8.32 whenever n is 1. Continuing in this way, the system arrives at the ideal gas law when the fourth level of description is reached.

In determining when to generate a new description to summarize a set of lower level descriptions, BACON.4 draws on a generalized version of the traditional inductive inference rule. This heuristic looks for recurring values of a dependent variable. Upon finding an invariance, it formulates an hypothesis about when the value will occur, including as conditions those independent values which were common among the observed examples. BACON.4 has primitive facilities for ignoring small amounts of noise in numerical data, but cannot deal with significant deviations from regularity.

Defining Theoretical Terms. In the ideal gas example given above, the dependent terms (V, pV, pV/T, pV/nT) about which generalizations were made became progressively more complex. Such combinations of directly observable variables may be viewed as a type of theoretical term, a term that is not directly observable but whose values are computable from observables. Although a term like pV/nT may be replaced by its definition at any time, its use can simplify the statement of a complex law considerably. How does the program arrive at useful theoretical terms such as pV/nT?

BACON.4 uses a heuristic search method to explore the space of theoretical terms. The program's heuristics note increasing and decreasing monotonic relations between pairs of variables that take on numeric values. If the slope is constant, then the system creates two new theoretical terms defined as linear combinations of the related variables. If the slope varies (the relation is not linear), then BACON.4 computes the product or ratio of the related terms and treats this variable as a new theoretical term. Once a new term has been defined, no distinction is made between it and directly observed dependent variables. BACON.4's ability to recursively apply the same heuristics to progressively more complex terms gives it considerable power in searching for empirical laws.

Postulating Intrinsic Properties. Although BACON.4's term detectors are useful for relating numeric variables, they do little good when an independent nominal or symbolic variable influences a numeric dependent term -- for example, when inserting different wires into a circuit alters the current. In such cases, the program calls on a heuristic for postulating an intrinsic property of the nominal variable (such as conductance). The values of this intrinsic property (a new theoretical term) are set equal to the values of the numeric dependent term, and each value is associated with the conditions under which the observation was made. The intrinsic values are retrieved whenever these conditions are met.

Upon defining an intrinsic property and specifying its values, BACON.4 also defines a new variable which is the ratio of the dependent term and the intrinsic property. This ratio, which we call a conjectured property, is guaranteed to be 1.0 for the observations that led to its postulation. However, if the intrinsic values are later retrieved in new situations, where the independent variables have different values, the conjectured property can take on different values, and may enter into new empirical laws.

In addition, whenever BACON.4 is about to assign a set of intrinsic values, it examines those values to see if they have a common divisor. If a common divisor is found, then the values are divided by this number and the resulting integers are associated with the nominal values instead. When the ratio of the dependent term and the intrinsic property is computed, this will equal the divisor rather than 1.0. Since different divisors may be found in different circumstances, BACON.4 may be able to relate this ratio to other terms even though the intrinsic values are never retrieved.

The Discoveries of BACON.4. BACON.4 differs from its precursor, BACON.3, mainly in its ability to propose intrinsic properties. However, even without this heuristic, the earlier system [1] was able to discover versions of the ideal gas law, Kepler's third law, Coulomb's law, Ohm's law, and Galileo's laws for the pendulum and constant acceleration. Drawing on its techniques for proposing intrinsic properties, BACON.4 has rediscovered a number of additional laws from the history of science. These include Snell's law of refraction [sine i/sine r = n(1)/n(2)], the law of conservation of momentum [m(1)v(1) = m(2)v(2)], the law of gravitation [$F = Gm(1)m(2)/d^2$], and Black's specific heat law [c(1)m(1)t(1) + c(2)m(2)t(2) = [c(1)m(1) + c(2)m(2)]t(f)]. The terms n, m, and c in these equations are intrinsic properties invented by the program.

More recently, BACON.4 has induced a number of the laws of Nineteenth Century chemistry from data on chemical reactions. In examining these reactions, the program treats three variables as independent: the element contributing to the reaction, the reactant compound, and the weight of the element used, w(e). Three dependent variables are measured: the weight of the reactant compound, w(c), the (gaseous) volume of the element used,

and the (gaseous) volume of the compound, v(c), under standard conditions. At the first level, the system finds three constant ratios for each element/compound combination: w(e)/w(c), w(e)/v(e), and w(e)/v(c). The first of these is the relative combining weight of the element, which is constant by Proust's Law. The second is the constant density of the element, and the third is the weight of the element per unit volume of the compound. At the second level, BACON.4 varies the compound and discovers that the w(e)/v(c) values occur in small integer multiples of each other. Using the common divisors discovered in this way as intrinsic properties of the elements, the program finds a common divisor among these latter numbers, thus obtaining the atomic weights of the elements as multiples of the weight of hydrogen.

The diversity of these laws suggests that BACON.4, and the heuristics that it employs, are in fact quite general. Thus, the notions of invariance, theoretical terms, intrinsic properties, and common divisors are applicable to a number of domains, and any future models of data-driven discovery should incorporate these concepts. Our future research will attempt to extend these techniques to discovery in other areas, and focus on the problems of noise and determining relevant variables.

Support:

This research was supported by NSF Grants SPI-7914852 and IST-7918266, NIMH Grant MH-07722, and ARPA Grant F33615-78-C-1551.

References:

[1] Langley, P. Rediscovering physics with BACON.3. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 1979, 505-507.

[2] Bradshaw, G. L., Langley, P., and Simon, H. A. BACON.4: The discovery of intrinsic properties. Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence, 1980, 19-25.

[Lopez de Mantaras] LEARNING THE MEANING OF IMPRECISE CONCEPTS

Personnel: R. Lopez de Mantaras, (IKERLAN, Apartado 146, Mondragon (guipuzcoa), Spain); C. Freksa, (EECS Dept., Univ. of California Berkeley, CA 94720); J. Aguilar Martin, (LAAS du CNRS, 7, Av. du Colonel Roche, 31400 Toulouse, France).

Objectives and methods:

We attempt to build a system capable of learning the meaning of imprecise concepts by means of a question-answering dialog, in English, with a human teacher. The teacher asks the system to identify objects, in a given context, described by its colors, sizes and shapes, this description is automatically translated into PRUF by a translator based on an Augmented Transition Network. The system answers with the object that fits the better the PRUF expression, this answer can or cannot be correct. This starts a dialog between the human and the system, this dialog allows the system to modify the possibility distributions associated with the known concepts (colors, size, etc.) and also to initialize new possibility distributions corresponding to new concepts. We consider that the system has learned the concepts when it can identify any object in the context, described by means of this concept.

Results and future plans:

The translator(2) has been successfully implemented in LISP, and now we are working on the Learning(3) and the Identification(8) processes and we intend to implement them in L-FUZZY(7).

References:

(1) ZADEH, L.A. (1978), "PRUF-A Meaning Representation Language for Natural Languages", Int. J. Of Man-Machine Studies, 10, 395-460.

(2) LOPEZ DE MANTARAS, R (1980), "Translation of Natural Language Descriptions of Simple Objects into PRUF", accepted for presentation in the Symposium of Artificial Intelligence and Cognitive Processes, XXII International Congress of Psychology, Leipzig (GDR)

(3) LOPEZ DE MANTARAS, R.,AGUILAR MARTIN, J. (1980), "Algorisme d'Adaptacio Recursiva de Funcions de Possibilitat per l'Apprecentatge de conceptes imprecisos", submitted to CIL 81, Barcelona (Spain)

(4) COGUEN, J.A. (1975), "On Fuzzy Robot Planning", Fuzzy Sets and their applications to Cognitive and Decision Processes, (edited by Zadeh, Fu, Tanaka and Shimura), Academic Press Inc., New York, 429-447.

(5) SHAFER, G. (1976), "A Mathematical Theory of Evidence", Princeton University Press.

(6) ZADEH, L.A. (1979), "On the Validity of Dempster's Rule of Combination of Evidence", Memorandum ERL, University of California, Berkeley.

(7) FREKSA, C. (1980), "L-FUZZY, An AI Language with Linguistic Modification of Patterns", Memorandum UCB/ERL M80/10, University of California, Berkeley.

(8) FREKSA, C. (1980), "Communication about Visual Patterns by means of Fuzzy Characterizations", Symposium of Artificial Intelligence and Cognitive Processes, XXII International Congress of Psychology, Leipzig (GDR).

[Michalski] INDUCTIVE LEARNING AND CONCEPTUAL DATA ANALYSIS

Personnel: Ryszard S. Michalski, Robert Stepp, John Davis, Albert Boulanger, Paul O'Rorke. Department of Computer Science, University of Illinois, Urbana 61801. Phone: (217) 333-6725. ARPA address: MICHALSK at DTI.

Objectives, methods, results

This project is concerned with the development of theory, efficient algorithms and experimental programs for inductive inference and conceptual analysis of data (pattern discovery). The developed programs are applied to selected practical problems, mainly in the area of agriculture and medicine.

Other research topics include deductive inference from uncertain premises and a formalization of inference rules which people use in answering questions for which they do not have a stored answer, but have the knowledge of relevant facts. (This work is in collaboration with an experimental psychologist, Alan Collins from Bolt, Beranek and Newman, Inc.).

The essence of learning is constructing, using and improving descriptions. We are interested specifically in symbolic descriptions, which are formulated in logic style notation, but closely related to natural language expressions (which humans might produce when

characterizing the same phenomena).

The choice of the language for formulating descriptions and of the allowed operators for modifying them is viewed as fundamental for successful research in machine learning. The language and the operators should not be too simple - so that the problem is sufficiently interesting, but not too rich - so that the research will not be impaired by the exponentially growing complexity. As understanding of learning processes improves, more advanced languages and operators are used. This progress has been reflected in our research by moving from initially studied propositional style descriptions (which used variable-valued logic language VL1) to predicate calculus style descriptions with typed variables and additional syntactic forms (language VL2).

Specific problems and results from our past and current research include:

1. learning propositional style descriptions from examples (results: a family of AQVAL programs [1-5,5.7,9])
2. learning structural descriptions from examples (results: programs INDUCE [6,7,14])
3. selecting most relevant descriptors (variables) from a large set of candidates (result: program AQPLUS implementing an adaptive random search)
4. generating new descriptors in the process of learning (i.e., problem of 'constructive induction') (some results incorporated in INDUCE 1.2 [14,15])
5. discovering rules characterizing sequences of objects (using the inductive card game Eleusis as a model problem) (result: program Eleusis [13])
6. revealing a conceptual hierarchical structure in data ('learning from observation') (result: program CLUSTER/paf [15,17])

A brief description of the two above mentioned programs, INDUCE 1.2 and CLUSTER/paf follows.

Program INDUCE 1.2 This program performs a structural inductive inference in which both the input examples and the output results are structural descriptions involving object attributes and relations among the object components and subcomponents. The program can accept some amount of 'world knowledge' to guide the induction process, and has a limited ability for 'constructive induction', i.e., for generating and using new variables, not provided in the input data. The underlying description language is VL2 - an extension of the first order predicate calculus, which uses typed variables and additional operators such as selector and internal disjunction.

Program CLUSTER/paf This program does what we call a 'conceptual clustering'. Given a collection of objects (observations, measurements, etc.), the program constructs a hierarchical structure of subcategories of objects, such that each subcategory is described (after an appropriate generalization) by a single conjunctive statement involving attributes of objects. The attributes may be nominal variables or relations on numerical variables. The structure is built in such a way, that a flexibly defined cost of a collection of descriptions branching from any node is minimized (the 'cost' may reflect the computational complexity of descriptions, the number of attributes, etc.)

Developed programs have been experimentally tested (by the authors and also by external users) on problems in several domains (medicine, plant pathology, chess end games, musicology, geology, texture analysis, and industrial product development). One of the most notable results was a determination of decision rules for soybean disease diagnosis by inductive learning from a few hundred examples of diseases supplied by plant pathologists [12]. These inductively derived rules gave a better performance (on sever hundred testing cases) than rules obtained by a formalization c plant pathologist's decision procedures.

The theoretical and practical results of the research have bee described in approximately 60 publications (a selection of them given in the Bibliography below).

Future Plans:

Among topics for future research are: an extension of th methodology of structural learning implemented in INDUCI methods of constructive induction, detecting regularities sequential processes, revealing a conceptual structure and discove of patterns in data, implementing learning in an expert system, ar formalization of human rules of plausible reasoning (in collaboratic with Alan Collins).

Support:

National Science Foundation Grant No. MCS 79-06614 ar U.S. Department of Agriculture Grant No. 321512344

References:

(a selection from approximately 60 papers written)

[1] R. S. Michalski, "A Variable-Valued Logic System Applied to Picture Description and Recognition," Chapter in t book, Graphic Languages edt. F. Nake and A. Rosenfel North-Holland Publishers, 1972.

[2] R. S. Michalski, "Discovering Classification Rules by t Variable-Valued Logic System VL1," Proceedings of the Thi International Joint Conference on Artificial Intelligence Stanfo California, August 20-24, 1973.

[3] R. S. Michalski, "AQVAL/1--Computer Implementation a Variable-Valued Logic system and the Application to Patte Recognition," Proceedings of the First International Jo Conference on Pattern Recognition Washington, D.C., Octol 30-November 31, 1974.

[4] R. S. Michalski, "Synthesis of Optimal and Quasi-Optir Variable-Valued Logic Formulas," Proceedings of the International Symposium on Multiple Logic Bloomington, India May 13-16, 1975.

[5] R. S. Michalski, "Variable-Valued Logic and Applications to Pattern Recognition and Machine Learnin Chapter in the monograph: MULTIPLE LOGIC AND COMPUT SCIENCE edt. David Rine, North-Holland Publishers, 1975.

[6] James B. Larson, "Inductive Inference in Variable-Valued Predicate Logic Systems VL2.1: Methodology Computer Implementation," Ph.D. Thesis, Report No. 8 Department of Computer Science, University of Illinois, Urba May 1977.

[7] R. S. Michalski, "A System of Programs Computer-Aided Induction: A Summary," Proceedings of the F International Joint conference on Artificial Intelligence M Boston, August 1977.

[8] R. S. Michalski and Pericles Negri, "An Experiment Inductive Learning in Chess End Games: The King-Pawn-K Case," Chapter in the book, Machine Representation of Knowle

MACHINE INTELLIGENCE 8 pp. 175-192, Ellis Horwood Ltd., 1977.

[9] R. S. Michalski and J. B. Larson, "SELECTION OF MOST REPRESENTATIVE TRAINING EXAMPLES AND INCREMENTAL GENERATION OF VL1 HYPOTHESES: the underlying methodology and the description of programs ESEL and AQ11," Report No. 867, Department of Computer Science, University of Illinois, Urbana, May 1978.

[10] T. G. Dietterich and R. S. Michalski, "Learning and Generalization of Structural Descriptions: Evaluation Criteria and Comparative Review of Selected Methods," IJCAI-79, Proceedings of the 6th International Joint Conference on Artificial Intelligence, p.223-231, Vol.1, Tokyo, August 20-23 1979 (an extended version to appear in Artificial Intelligence J. 1981).

[11] R. Stepp, "Learning Without Negative Examples via Variable-Valued Logic Characterizations: The Uniclass Inductive Program AQ7UNI," Report No. 982, Department of Computer Science, University of Illinois, July, 1979.

[12] R. S. Michalski and R. L. Chilausky, "Learning By Being Told and Learning From Examples: an experimental comparison of two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis," A Special Issue on Knowledge Acquisition and Induction, Policy Analysis and Information systems, No. 2, 1980.

[13] T. D. Dietterich, "The Methodology of Knowledge Layers for Inducing Descriptions of Sequentially Ordered Events," M.S. Thesis and Report No. 1024, Department of Computer Science, University of Illinois, May 1980.

[14] R. S. Michalski, Pattern Recognition as Rule-Guided Inductive Inference, IEEE Trans. on Pattern Analysis and Machine Learning, vol. PAMI-2 No.4, July 1980.

[15] R. S. Michalski, "KNOWLEDGE ACQUISITION THROUGH CONCEPTUAL CLUSTERING: A Theoretical Framework and an Algorithm for Partitioning Data Into Conjunctive Concepts," A Special Issue of Knowledge Acquisition and Induction, Policy Analysis and Information Systems, No. 3, Sept. 1980.

[16] R. S. Michalski, "Inductive Learning as Rule-Guided Generalization and Conceptual Simplification of Symbolic Descriptions: Unifying Principles and a Methodology," Workshop on Current Developments in Machine Learning, Carnegie-Mellon University, Pittsburgh, July 16-18, 1980.

[17] R. S. Michalski and R. Stepp, "Revealing a Conceptual Structure in Data by Inductive Inference," Invited Paper for the 10th International Machine Intelligence Workshop, Case Western Reserve University, April 20-25, 1981.

[Mitchell] LE ARNING PROBLEM-SOLVING HEURISTICS BY EXPERIMENTATION

Personnel: Tom M. Mitchell, Paul E. Utgoff, Bernard Nudel (Computer Science Department, Rutgers University, New Brunswick, NJ, 08903, USA), Ranan B. Banerji (Computer Science Department, Temple University, Philadelphia, PA, USA)

Objectives and methods:

Our objective is to develop methods by which heuristic problem-solving programs can improve their performance through practice. Our current research has two major thrusts: (1) to design and analyze general methods for learning problem-solving heuristics by experimentation, and (2) to construct a computer program to test our ideas empirically in a particular task domain.

Our initial system to learn heuristics, called LEX, is being developed using the task domain of symbolic integration. LEX starts with a set of operators for symbolic integration, similar to the operators found in freshman calculus books (e.g., integration by parts). LEX must learn heuristics that control the application of these operators within a forward heuristic search problem solver. Each such heuristic constitutes a specialization of one of the given operators, and is inferred from observations of when that operator is useful and when not. For example, the following heuristic is representative of those that LEX has learned:

IF the integrand is the product of x with
any transcendental function of x

THEN try Integration by Parts,
(e.g., INTEGRAL u dv => uv - INTEGRAL v du)
with u bound to x, and dv bound to transcendental(x) dx.

(note: a transcendental function is any trigonometric, exponential, or logarithmic function)

The design of LEX involves four distinct modules: (1) a PROBLEM-SOLVER that conducts a forward heuristic search, (2) a CRITIC that analyzes the search conducted by the PROBLEM-SOLVER, in order to extract training examples of appropriate and inappropriate search steps performed in attempting to solve the problem, (3) a GENERALIZER that infers general heuristics from the specific training examples supplied by the CRITIC, and (4) a PROBLEM-GENERATOR that produces practice problems. This organization is similar to that described in [1].

The initial design of LEX is detailed in [2]. The GENERALIZER uses the version space generalization method [3], [4], originally developed as part of the Meta-DENDRAL program. Briefly, we define the version space of a heuristic with respect to a set of training examples of the heuristic, and with respect to a given language in which to state heuristics, as the set of *all* heuristics describable in the given language that are consistent with these training examples. The GENERALIZER initializes, then refines the version space of each heuristic, by examining the sequence of training instances derived from solved practice problems. At each step, the GENERALIZER efficiently represents the version space of each heuristic by storing the maximally specific and the maximally general descriptions (versions) of the heuristic that are consistent with observed data.

Representing partially learned heuristics in terms of their version space of alternative descriptions allows the PROBLEM-SOLVER to utilize tentatively formulated heuristics in a reasonable way in directing its search to solve subsequent practice problems. It also provides one important tactic for the PROBLEM-GENERATOR to suggest useful practice problems; that is, to generate problems that match some but not all descriptions in the version space of a given heuristic.

Results:

We have implemented the PROBLEM-SOLVER, CRITIC, and GENERALIZER modules of LEX, and have begun debugging and experimenting with the system. The current system has approximately 20 intial operators, and is able to learn heuristics for many of them by solving and analyzing practice problems that we present by hand. We hope to present the results of preliminary experiments with the system at IJCAI7 this August [5].

Future Plans:

Our research agenda includes the following items:

- Try to increase the power of the LEX by allowing it to reason about problem solutions. We believe that the ability to "understand" why a particular operator worked in a particular instance should lead to more powerful methods for formulating heuristics.
- Consider how LEX might detect that its current language for describing heuristics is insufficient, and how it might repair its language in such cases.
- Experiment with alternative startegies for automatically generating practice problems.

Support:

We thank the National Science Foundation (grant MCS80-08889) and the National Institutes of Health (grant RR-643-09) for current and past support of this research.

References:

1. Smith, R.G., et al, A Model for Learning Systems, Proceedings 5th International Joint Conference on Artificial Intelligence, 1977.

2. Mitchell, T. M., Utgoff, P. E., and Banerji, R. B., Learning Problem-Solving Heuristics by Experimentation, Proceedings of the Workshop on Machine Learning, Carnegie-Mellon University, July, 1980.

3. Mitchell, T.M., Version Spaces: An Approach to Concept Learning, Ph.D. dissertation, Stanford Univ., December 1978. Also Stanford Computer Science report STAN-CS-78-711, HPP-79-2.

4. Mitchell, T.M., Generalization as a Search Problem, Proceedings 6th International Joint Conference on Artificial Intelligence, Tokyo, Japan, 1979.

5. Mitchell, T.M., Utgoff, P.E., Nudel, B., and Banerji, R.B., Learning Problem Solving Heuristics from Practice, submitted to IJCAI7, 1981.

[Mostow] MACHINE-AIDED HEURISTIC PROGRAMMING: LEARNING BY BEING TOLD

Personnel: Jack Mostow. This summary describes my dissertation research, performed at Carnegie-Mellon University Computer Science Department. My thesis committee consists of F. Hayes-Roth (supervisor), A. Newell, J. Carbonell, and B. Balzer. The research fits into Hayes-Roth's knowledge acquisition work at the Rand Corporation.

Maior objectives and methods:

My thesis lies within the paradigm of *machine-aided heuristic programming*, or *learning by being told*. The idea is to instruct a computer to do a new task in much the same way one would instruct a person: by giving an informal description of the task plus some advice for how to do it well, expressed in the terms of the task domain (rather than in a fixed programming language). For example, if the task is to play the card game Hearts (one of the tasks we've used as an experimental vehicle), the task description would include the rules of the game, and the advice would include heuristics like "avoid taking points" and "don't lead a suit in which an opponent is void." It would also include definitions of domain concepts, e.g., "a player with no cards in a suit is said to be void in that suit."

Converting this kind of task description into effective behav poses several problems: *encoding* an informal natural langu description into a precise internal representation of its meani *operationalizing* it in terms of basic capabilities, *integrating* differ pieces of advice, and *applying* knowledge to specific task situations runtime.

My thesis focuses on operationalization: the conversion knowledge into a form effectively applicable to a task. T operationality of advice depends on the information a computational capabilities available to the agent performing the ta To limit the problem, my thesis addresses a restricted (but s quite broad) version of operationalization: *the transformation well-defined expressions into expressions that will (frequently) be feas executable by a specified mechanism using the data and procedu available to it at runtime.* This problem can be characterized *learning by devising procedures to do what you're told.*

Another goal of this work was to encode knowledge about in a form suitable for mechanical application. What methods used in AI? What reasoning is required to apply them to a giv problem? How can general methods use knowledge about particular task domain?

Results:

To explore these questions, I built an interactive program operationalize an input expression. The key features of the syst are as follows:

1. Advice and definitions of domain concepts are input in an unambiguous LISP-like language (to bypass issues of natural language understanding).
2. Advice is interactively operationalized by a series of transformations performed by *general rules* that access a base of *domain knowledge*. At each point in this process, I select a rule and the system applies it to the current expression to produce a transformed expression.
3. My system operationalizes well-defined advice by figuring out how to *achieve* a desired condition or *evaluate* a specified quantity. For example, the program operationalizes "avoid taking points" as "play a low card." It also constructs an alternative operationalization in terms of heuristic search, by formulating a search space of card sequences for the trick and using general refinement rules to prune and order the search. It operationalizes the problem of deciding whether an opponent is void in several ways: by reasoning that an opponent who was void earlier in the round is still void; by deducing from the rules of the game that a player who breaks suit is void in the suit led; and by applying a general formula for the probability that two subsets chosen randomly from a given set will be disjoint.

To test the generality of the system, I used it to operationali a simple music composition task in terms of heuristic search.

Future plans:

I have just joined Stanford University's Heurist Programming Project as a Research Affiliate, and plan to work wi Doug Lenat. I hope to apply some of the ideas in my thesis to mo respectable task domains, embedding my representation in RLL exploit its built-in inheritance and caching mechanisms f efficiency.

Support

My thesis research has been supported by CMU Computer Science Department and at the Rand Corporation under Grant MCS77-03273 from NSF.

References

Mostow, D. J. Mechanical operationalization of concepts and heuristics (working title). Ph.D. thesis. Carnegie-Mellon University Computer Science Department, Pittsburgh, PA. In progress.

Mostow, D. J. Operationalization in terms of heuristic search (working title). To appear in Proceedings of the Workshop on Machine Learning. In progress.

Hayes-Roth, F., Klahr, P., and Mostow, D. J. Advice-taking and knowledge refinement: an iterative view of skill acquisition. P-6517. The Rand Corporation, Santa Monica, CA. July 1980.

Hayes-Roth, F., Klahr, P., and Mostow, D. J. Knowledge acquisition, knowledge programming, and knowledge refinement. R-2540-NSF. The Rand Corporation. May 1980.

Mostow, D. J., and Hayes-Roth, F. Machine-aided heuristic programming: a paradigm for knowledge engineering. N-1007-NSF. The Rand Corporation. February 1979.

Hayes-Roth, F., Klahr, P., Burge, J., and Mostow, D. J. Machine methods for acquiring, learning, and applying knowledge. P-6241. The Rand Corporation. October 1978.

[Neches] HEURISTIC PROCEDURE MODIFICATION

Personnel: Robert Neches, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, PA 15260.

Objectives and methods:

This project explores processes by which learners acquire and improve procedures for solving real-world problems. In particular, it is concerned with developing and testing a model of "strategy transformations" (Neches & Hayes, 1978; Neches, 1979), modifications to a procedure driven by experience with its execution.

Within this model, learning heuristics are based upon a taxonomy of strategy transformation types (e.g., deleting unnecessary parts, saving partial results, re-ordering the sequence of operations). For each transformation type, there are heuristics for when to set up goals to make that kind of change to a procedure. These heuristics are stated as production rules conditioned upon various configurations in the goal trace.

The primary theme underlying this research is that of exploring how meta-knowledge can be exploited in an intelligent learning system. In one way or another, the key to all of the mechanisms described below is that they define a representation for some class of processing information, establish criteria for interesting configurations in that representational structure, and then specify actions to be taken in response to instances of those configurations. I regard my attempts to build a system which uses meta-knowledge about knowledge representations as an attempt to lay out a middle ground between knowledge-intensive expert systems and "syntactic" production learning models like Anderson's ACT.

There are two major thrusts to the research effort. The first is the development of a production system architecture for computer simulations of cognitive learning processes. The current version of this system, tentatively named NEWHPM (Neches, 1980), is an extension of earlier work on this topic in collaboration with Pat Langley (Langley, Neches, Neves, & Anzai, 1980). The second major thrust is the implementation of a strategy transforming system as a set of productions residing within NEWHPM (see also Neches, 1980).

NEWHPM is a production system for computer simulations of cognitive learning processes. The system reflects a philosophy of representing learning as a process of piece-wise rule acquisition, in which the space of "useful" rules is only a small portion of the space of potential rules. NEWHPM provides basic facilities for developing correct versions of new rules (obtained from Pat Langley's ACTG), along with hooks which allow users to implement heuristics for directing the system's attention to interesting candidate rules. Knowledge is represented as a semantic network in which propositions are defined by sets of node-relation-node triples. Each triple has an activation level associated with it; productions fire upon matching data above a minimum threshold.

Among the features distinguishing NEWHPM from related production systems are: (a) PRODUCTION TRACES -- copies of each production instantiation are added to the network after each cycle, making them available to user-defined learning productions; (b) GOAL TRACES -- a set of representational conventions for procedures as hierarchical goal structures which, in conjunction with the production trace, makes the system's memory for its actions both goal-ordered and time-ordered; (c) GOAL-DRIVEN SPREADING ACTIVATION -- a capacity (which, to my knowledge, no other psychological simulation has) for making the spread of activation through the semantic network dependent on the kind of processing currently being done (for example, activation is sent downwards in the goal structure when a goal is initiated, but upwards and sideways when a goal is terminated); (d) CONFLICT RESOLUTION BY CLASSES -- productions fire in parallel in NEWHPM, and are grouped into classes which are each handled by a separate user-defined conflict resolver that can access arbitrary "production-descriptions" in making its selections; (e) ISOMORPHIC STRUCTURE MATCHING -- NEWHPM allows productions to find matches between network structures which are isomorphic, but not identical, thus permitting user-defined learning processes to find and compare instances of related events; (f) AUTOMATIC EFFORT ESTIMATION -- the system updates records in the network estimating the amount of effort put into processing each active goal.

One use of the NEWHPM production system architecture is in my efforts to get a computer to emulate some of the discoveries which small children make about basic arithmetic procedures. Young children (approx. 4 years) are frequently observed adding by a counting procedure called the "SUM method". This method consists of (a) counting out a set of objects to represent the first addend; (b) counting out another set of objects to represent the second addend; (c) merging the two sets and counting the number of objects in the new set. Within a few years, most children appear to have adopted a host of specialized strategies, the most prevalent of which is called the "MIN method". In this very different method, the child starts with the larger number and increments it the number of times given by the smaller addend.

In previous work, I have shown that this change can be simulated in an OPS2 production system. In that simulation, a production system for the SUM method was successively modified into closer and closer approximations of the MIN method by the addition of hand-written productions. Productions were added one at a time, with each addition yielding a working strategy that could be shown to have some psychological plausibility as an intermediate stage in the learning process.

I am now engaged in trying to get a production system, BASICS.HPM, to automate the process of acquiring the new productions. BASICS.HPM runs in my own production system architecture, NEWHPM, and modifies productions contained in a simulation of the SUM method called ADD.HPM. The addition simulation builds an hierarchical goal structure in order to perform its task. BASICS.HPM contains heuristics which look for configurations in goal structures, and suggest changes appropriate to those configurations.

The process of the system learning the MIN procedure consists of these heuristics applying in discoveries such as that: (a) generating the first addend's set is redundant with counting the first part of the total set, leading to a strategy of continuing the total count from the count of the first addend; (b) the count of the first addend contains unused results when only the last number in the count is considered, leading to a strategy of counting on from one of the addends; (c) when the strategy of counting on from an addend is used, faster results are correlated with cases where the strategy started with the larger addend rather than the smaller.

As in the design of the NEWHPM architecture itself, the theme which I am developing in BASICS.HPM is that of exploiting meta-knowledge in order to guide a learning system. The heuristics of looking for matching sub-goal structures, unconnected goal nodes, and correlates of effort differences, all depend on knowing what constitutes interesting data configurations in a given knowledge representation.

Results:

The NEWHPM architecture is implemented and debugged, with the exception of the code for automatic effort estimation. Several production systems have been implemented and run on NEWHPM, including the ADD.HPM simulation of children's addition strategies described above.

As of October 31, 1980, the strategy transformation heuristics described above have been coded, but have not yet been fully debugged.

References:

Langley, P.W., Neches, R., Neves, D., and Anzai, Y. A domain-independent framework for procedure learning. JOURNAL OF POLICY ANALYSIS AND INFORMATION SYSTEMS, 4(2), 163-197, 1980.

Neches, R. HEURISTIC PROCEDURE MODIFICATION. Pittsburgh, PA: Psychology Department, Carnegie-Mellon University, unpublished Ph.D. thesis in preparation, 1980.

Neches, R. PROMOTING SELF-DISCOVERY OF IMPROVED STRATEGIES (CIP #398). Pittsburgh, PA: Psychology Department, Carnegie-Mellon University, technical report, 1979.

Neches, R. & Hayes, J.R. Progress towards a taxonomy of strategy transformations. In A.M. Lesgold, J.W. Pellegrino, S. Fokkema, & R. Glaser (Eds.), COGNITIVE PSYCHOLOGY AND INSTRUCTION. New York: Plenum Books, 1978.

[Novak] COGNITIVE PROCESSES AND KNOWLEDGE STRUCTURES USED IN SOLVING PHYSICS PROBLEMS

Personnel: Gordon S. Novak Jr., Agustin A. Arayam, Man-Lee Wan. Computer Science Department, University of Texas at Austin, Austin, Texas 78712

Objectives, methods, and results:

The goal of this project is to investigate the cognitive processes and knowledge structures needed for expert-level problem solving in physics. We are doing this by analyzing protocols of novice and expert humans solving physics problems, by writing a program which can solve actual problems covering a variety of physical principles, and by investigating how a program can learn expertise in solving physics problems through practice. The latter program, which is currently being implemented, is the one we will discuss here.

We believe that the human who is expert at solving physics problems does not apply the laws of physics in a deductive fashion to solve problems [indeed, this is impossible, since real problems involve infinitely many insignificant factors which cannot be taken into account]; instead, the expert problem solver recognizes a large number of special classes of problems, each of which is associated with a simplified method for solving problems in that class. For example, to answer a simple question like "What force is required to lift one end of a 40 lb pole?", the expert will recognize immediately that the answer is half the weight of the pole; this special-case method is clearly easier than writing and solving the six simultaneous equations which govern static equilibrium in three dimensions.

Our model of the learning situation is as follows. We assume that the problem solver begins with a general formulation of a physical law, as is typically found in physics textbooks; in the case of rigid body statics, the general formulation would be the six equations of static equilibrium. The problem solver solves a problem using its current solution method, keeping a protocol of its steps as it proceeds. After the problem has been solved, the protocol is analyzed to find "interesting" features which can lead to a more economical solution process for problems of that type. Examples of such "interesting" features include terms which are zero (e.g., initial velocity of a body accelerating from rest), equations which were written but were not used in finding the desired unknowns (e.g., horizontal force equations in a problem in which all significant forces are vertical), factors which are much smaller than the significant factors and can thus be ignored (e.g., the weight of a ladder is typically much less than the loads it supports), and nonlinear equations which are "nearly" linear. Given such an interesting feature, the problem-solving process is modified to solve such a problem more efficiently. A discrimination net is used to determine the problem type and associated solution method; to modify the problem solving process, a test is added to the discrimination net above the point at which the more general solution method was recognized, and a simplified solution method constructed from the previous solution method is used when the test is satisfied.

Support:

This research is supported by NSF Award No. SED-791280 in the Joint National Institute of Education - National Scienc Foundation Program of Research on Cognitive Processes and th Structure of Knowledge in Science and Mathematics.

References:

1. Novak, G. "Computer Understanding of Physics Problem Stated in Natural Language", American Journal of Computation Linguistics, Microfiche 53, 1976; Technical Report NL-3 Computer Science Dept., Univ. of Texas at Austin.

2. Novak, G. "Representations of Knowledge in a Progra for Solving Physics Problems", Proc. 5th IJCAI, Cambridge, Mas Aug. 1977, pp. 286-291.

3. Novak, G. and Araya, A. "Research on Expert Proble Solving in Physics", Proc. 1st AAAI Conference, Stanford, Cali Aug. 1980, pp. 178-180.

[Rychener] GRACEFUL EXAMPLARY PROGRAMMING

Personnel: Mike Rychener, Carnegie-Mellon University. Results may be applicable to work on expert systems with Allen Newell, John McDermott and Charles Forgy; and to work on cooperative user interfaces with Raj Reddy, Gene Ball and Phil Hayes.

Objectives and methods:

A system called GRAEP (Graceful Exemplary Programming) is being developed. GRAEP is to become an intelligent personalization agent within a user interface to a multi-media message system. That is, GRAEP will provide the capability for adapting to a user's idiosyncrasies, and will allow the user to more explicitly personalize various aspects of the system, including syntax, spelling, graphics, and repetitive sequences of system commands. The central organizing aspect of GRAEP is Exemplary Programming (EP), modelled after the work of Waterman, et al, at RAND. Under this EP approach, a user specifies some procedure to be performed by going through, step by step, an example of that procedure. Implemented as a production-rule system, it takes schemas as its main organizing data structure, and uses a form of semantic network for storing domain-dependent facts. All long-term knowledge in GRAEP is represented as rules, which includes the initial working system and the above-mentioned schemas and networks, which are acquired through interaction. Each slot in a schema is represented as a set of rules, the formats of which are usually specialized according to the functions performed by the particular slots. Each node in the semantic net, similarly, is a set of rules.

Results: Work on implementation is still in progress.

Support: ARPA.

References:

Rychener, Michael D., "A semantic network of production rules in a system for describing computer structures." Carnegie-Mellon University, Dept. of Computer Science, Tech. Report CMU-CS-79-130. June, 1979.

Rychener, Michael D., "Approaches to knowledge acquisition: the instructable production system project." Proc. First National Conference on Artificial Intelligence, pp. 228-230. August, 1980. Expanded version in preparation.

Rychener, Michael D., "Personalizing a user interface with rule-based exemplary programming." In preparation.

Waterman, D. A., Faught, W. S., Klahr, P., Rosenschein, S. J., and Wesson, R., "Design issues for exemplary programming." RAND note N-1484-RC. April, 1980.

[Schank] LEARNING PROJECTS AT THE YALE UNIVERSITY ARTIFICIAL INTELLIGENCE PROJECT

Personnel: Faculty: Roger Schank, Wendy Lehnert, Chris Riesbeck, Drew McDermott. Students: Mark Burstein, Gregg Collins, Janet Kolodner (now at Georgia Tech.), Michael Lebowitz (now at Columbia U.), Mallory Selfridge (now at U. Conn.).

Objectives and methods:

Our earliest learning project was language acquisition. Our newer projects are based on the MOP representation scheme for memory structures [Schank, *Cognitive Science*, *4*, 4].

We are concerned with two major topics: how does learning fit into the complete understanding process, and how does past knowledge (especially from reminding experiences) fit into learning. Noticing patterns in input texts and capturing the generalizations in reorganized memory structures is one class of projects. Remembering processing failures and being reminded of them when similar failures occur later is the other class.

a) Language acquisition (Selfridge) -- modelling child language acquisition from 1 to 2 years.

b and c) Generalization (Kolodner and Lebowitz) -- forming new MOPs as indexing structures for events as they are stored in memory.

d) Domain acquisition (Riesbeck, McDermott, Burstein, Collins) -- indexing rule failures, and constructing new MOPs, causal relationships, etc., during text understanding of a new domain.

e) Script acquisition (Burstein) -- forming new scripts, by debugging processing failures in existing scripts, using plan and goal knowledge (in MOP form).

Results -- Programs:

a) KID (Selfridge), using a simple world model (with knowledge about objects, like balls and tables, and what you do with them, and about people and the kinds of goals they have), learned from this context procedural definitions for understanding words like "get" and "put".

b) CYRUS (Kolodner) has a long-term, self-organizing memory for events reported in newspaper articles involving Cyrus Vance (and now Edmund Muskie). Reconstructive processes are used to retrieve those events from memory when answering questions.

c) IPP (Lebowitz) analyzes newspaper articles about international terrorist activities directly into its memory, making generalizing inferences as it does so.

d) ALFRED, in two prototypes, (Burstein and Collins) takes sequences of conceptual summaries of newspaper articles. Understanding failures cause debugging and reorganization of the processing structures.

e) This is a new project, springing from (c).

Future Plans:

Learning is a major project in many forms. CYRUS is being integrated with IPP and FRUMP (a newswire skimming and summarizing program -- DeJong, (1980) Yale Ph.D. Thesis) to produce a broader reading and learning system. The prototypes of ALFRED are being merged and the naive economic model (which is to change with experience) is being revamped.

Failure-driven reminding is at the heart of ALFRED and of a failure/explanation project (Schank and Collins) which is just beginn ing.

Support:

Advanced Research Projects Agency, the Office of Naval Research, and the National Science Foundation.

References:

Kolodner, J. Retrieval and organizational strategies in conceptual memory: A computer model. Ph. D. Thesis. (1980, forthcoming).

Lebowitz, M. Generalization and memory in an integrated understanding system. Ph. D. Thesis. (1980, forthcoming).

Selfridge, M. A process model of language acquisition. Ph. D. Thesis. Yale University, Research Report #172. (1980).

[Shapirol INDUCTIVE INFERENCE OF THEORIES FROM FACTS

Personnel: Ehud Shapiro, Yale University, Department of Computer Science

Objectives, methods, and results:

This paper is concerned with the inductive inference problem faced by a scientist, working in a given domain under some fixed conceptual framework, performing experiments and trying to find a theory capable of explaining their results. The Model Inference Problem is an abstraction of this setting, in which the domain of inquiry is some unknown model M for a given first order language L, experiments are tests of the truth of sentences of L in M, and the inductive inference problem is to find a set of true hypotheses that imply all the sentences tested and found true.

The main result of this paper is a general incremental inductive inference algorithm, based on the Popperian methodology of conjectures and refutations [Popper 59, Popper 68]. This algorithm can be shown to identify in the limit [Gold 67] any model in a family of complexity classes of models is most powerful of its kind, and is flexible enough to have been successfully implemented for several concrete domains. A system, based on the algorithm specialized to infer theories in Horn form, has been implemented in the programming language Prolog [Pereira et al. 78]. For example, in the domain of arithmetic, this system inferred the following set of axioms, from facts such as "0<0' is true", "plus(0',0',0'') is true", "times(0',0,0') is false", etc.

```
0<X
X<Y' _ X<Y

plus(0,X,X)
plus(X',Y,Z') _ plus(X,Y,Z)

times(0,X,0)
times(X',Y,Z) _ times(X,Y,W)^plus(W,Y,Z)
```

interpreting X' as the successor of X, plus(X,Y,Z) as X plus Y is Z, times(X,Y,Z) as X times Y is Z and P_Q^R as P follows from Q and R.

The system inferred these axioms in less than 30 seconds CPU time and from fewer than 50 facts. The system has also discovered an axiomatization for dense partial order with end points. It has successfully inferred logic programs (modulo their control component [Kowalski 79]) for simple list- processing tasks such as append, reverse and most of the other examples described in [Summers 76], and also logic programs for satisfiability of boolean formulas, binary tree inclusion, binary tree isomorphism, insertion sort and others.

The inference algorithm, called the Layers algorithm, has two tunable parameters: one determines how complicated the hypotheses are; the other how complex derivations from the hypotheses can be. Together these parameters determine the class of models that be inductively inferred in the limit by the algorithm. On the one hand they can be set so that the Layers algorithm can identify in the limit any model with complexity bounded by any fixed recursive function. On the other hand they can be set so that the Layers algorithm, appropriately implemented, can infer axiomatizations of concrete models from a relatively small number of facts in a practical amount of time, as the examples above show.

As part of the general algorithm, an algorithm for backtracing contradictions was discovered. This algorithm is applicable whenever a contradiction occurs between some conjectured theory and the facts. By testing a finite number of ground atoms for their truth in the model the algorithm can trace back a source for this contradiction, namely a false hypothesis, and demonstrate its falsity by constructing a counter- example to it.

Tests of the kind performed by this algorithm are known in philosophy of science as *crucial experiments.* Although their importance is recognized by most methodologies, an algorithmic way of sequencing them that *guarantees* singling out a false hypothesis is a novelty. The existence of such an algorithm apparently contradicts a claim of Duhem [Duhem 54], which simply denies its possibility. This claim was stated in support of what later came to be known as the Duhem-Quine thesis [Harding 76] of the irrefutability of scientific theories. The existence of the contradiction backtracing algorithm, and of a general inductive inference algorithm that incorporates it, may renew the philosophical discussion concerning the issue of refutability.

REFERENCES

E. Shapiro. Inductive inference of first order theories from facts. To appear as a Yale Computer Science Dept. Research Report No. 192.

Pierre Duhem. *The Aim and Structure of Physical Theory.* Princeton University Press, 1954. Originally published in French in 1906.

E. M. Gold. Language identification in the limit. *Information and Control* 10:447-474, 1967.

Sandra G. Harding (ed.). *Can Theories be Refuted? Essays on the Duhem-Quine Thesis.* D. Reidel Publishing Company, 1976.

Robert A. Kowalski. Algorithm = Logic + Control. CACM 22, July, 1979.

L. Pereira, F. Pereira and D. Warren. *User's Guide t DECsystem-10 PROLOG.* Technical Report 03/13/5570, Labortori Nacional De Engenharia Civil, Lisbon, September, 197 Provisional version.

Karl R. Popper. *The Logic of Scientific Discovery.* Basic Book New York, 1959.

Karl R. Popper. *Conjectures and refutations: The Growth Scientific Knowledge.* Harper Torch Books, New York, 1968.

Philip Dale Summers. *Program Construction from Exampl* PhD thesis, Yale University, 1976. Computer Science De research report No. 51.

[Sleeman] A RULE BASED MODELLING SYSTEM

Personnel: Derek Sleeman, University of Leeds (CMU, 80/81).

Objectives, methods, and results:

LMS(Leeds Modelling System) is a data-driven modelling system which has been used in several domains. LMS has to be provided with domain knowledge in the form of Production Rules, similarly with mal-rules which 'capture' difficulties which children have been noted to experience and with a set of problems which are sufficient to discriminate between deviant behavior. The system 'observes' the student's behavior on an example set and produces production system model(s) which, if executed, would produce the same solutions as the student.

LMS has recently been subjected to a 'test' in a Leeds High school; a close correlation between the 'bugs' diagnosed by LMS and those discovered during conventional diagnostic interviews was noted. (The subject domain was simple linear algebraic equations). This experiment clearly shows the diagnostic power of the system but did point to some shortcomings, it is planned to investigate these during the next session:

1. Reorganize the Modeller into off-line and on-line phases so as to increase the its speed (and improve its report writing facility)

2. Generate problems to discriminate between hypothesized models. Use the rules (including the mal-rules) to generate problems or problem-templates which would be sufficient to discriminate between a pair of hypothesized models.

3. Cope with the situation where the appropriate Mal-rules are NOT provided. 3 approaches will be considered:

A. Analyze protocols 'manually' as at present and then subsequently enhance the list of mal-rules.

B. Generate mal-domain rules from domain rules given constraints for the syntax and semantics of mal-rules . (One could argue that this has some psychological validity).

C. Use a 'standard inference' technique eg Version space, to refine a domain rule given that using it the student got some problems right and others wrong. The essential problem here is to determine WHICH rule(s) are causing the difficulty; and so one needs to either: use an inference method which can accommodate ERRORFUL data, or probe the student, as a good teacher would, to find out more exactly the nature of his difficulties with each incorrectly worked problem. (Indeed one might also profitably do that for some of the correctly worked problems too).

4. Simulate the processes by which a student who can 'do' arithmetic generalizes this competence to cope with Algebra.

References:

1. D.H. Sleeman, 1979. Some current topics in Intelligent Teaching Systems, AISB Quarterly, 33, pp22-27.

2 D.H. Sleeman and M.J. Smith, Modelling Student's Problem Solving, AI Journal, (to appear).

3 D.H. Sleeman (1981) Assessing aspects of competence in Basic Algebra , in Intelligent Tutoring Systems, ed D.H. Sleeman and J.S. Brown, New York: Academic Press.

[Srinivasan] KNOWLEDGE BASED LEARNING SYSTEMS

Personnel: Chitoor .V. Srinivasan, David M. Sandford, Richard Keller, Donna Nagel (Department of Computer Science, Rutgers University, New Brunswick, N.J. 08903).

Objectives and Methods:

This research is concerned with acquisition and utilization of "domain knowledge" for problem solving, and techniques for implementing "knowledge based systems". There are four aspects to our work: The first pertains to fundamental studies on knowledge representation, the logical foundations of a meta theory of systems that represent and use knowledge. The second is investigation of methods by which a system can automatically learn this knowledge from examination of examples of states that occur in a domain. The third is the use of domain knowledge to construct procedures that solve certain kinds of "goal satisfaction" problems in the domain, and the fourth is investigation of methods by which models that capture domain knowledge can be used in a theorem proving system.

We view domain knowledge as consisting of three kinds of information: (1). Facts, (2). domain theories and (3). Methods for effectively using the facts and theories to state and solve problems. Depending on the way these three kinds of information are presented to the computer, we classify knowledge representation schemes into two categories: (i). Program based representations and (ii). Theory based representations. The schemes in the first category require that the specification of domain knowledge include also the specification of methods for using the knowledge. The methods are usually specified using "procedural attachment" to components of frames, scripts, structural inheritance nets, units, etc. In systems like these it is not possible to identify the "domain theory" (item 2 above). This effectively robs such systems of the ability to reflect about their own domain theories and use the theories to respond to unanticipated situations. All the existing "expert systems" and "language understanding systems" are based on "program based representations".

In "theory based" systems the domain theory is explicitly represented in the system as a declarative theory. The meta theory of systems of this kind specifies the methods needed to use the domain theory to state and solve problems. Thus in a first order system its meta-theory gives us the theorem proving methods to state and solve problem using the theory.

An alternative scheme of this kind that combines beautifully the "procedural" and "declarative" representations of a "domain theory" is the "logic programming" scheme, proposed by Kowalski. This uses a "theorem proving" control to interpret statements in "Horn clause" form as "programs". The programming paradigm used in "logic programming" provides a programmer facilities for specifying "control strategies" as a part of a representation. However, this system cannot function effectively in the context of incomplete knowledge. The full potential of this promising approach to knowledge representation is yet to be realized. Generating representations within this scheme is just as hard as generating representations in first order logic. The addition of facilities for specifying control strategies seems to make the specification even a bit harder.

A third approach to theory based representation is provided by the Meta Description System (MDS). In MDS we do not use "theorem proving" for problem solving. But a first order theory is interpreted in a standard model of the domain. This interpretation process is used as the basis for problem solving. The system is

intrinsically more tolerant of missing information (missing facts in its database or missing steps in its programs. The meta-theory of MDS shows how the interpretation process can be used to so a variety of problems in a domain. The problems include the standard kinds of goal satisfaction problems and also problems that require general theorem proving. A method of commonsense reasoning is defined, which is based on things called residues, which are logical expressions that explain why certain things are true in a state of the domain, others are false, and yet others are unknown. Problem solving methods involve, processes of extraction of residues from a given domain theory and a given state of the domain, using the residues to do means-end analysis, and methods for generalizing the results of a problem solving process.

The meta theory of MDS-like systems [Srinivasan,1980,1981] shows why, in principle, it is possible in such a system to "learn" domain theory from examination of examples of states in the domain, and also learn "domain heuristics" from problem solving experience in a domain. But we do not have viable implemented systems that can demonstrate these capabilities. We are now investigating possible "bootstrapping" techniques to implement a problem solving system based on MDS, in MDS itself. We have little hope of realizing "theory forming" in the foreseeable future. But we are conducting some very simple experiments to test the potential for "theory forming".

Use of Sophisticated Models in Theorem Proving.

"Model strategies" in resolution theorem proving have traditionally used only trivial models (models that could not realistically capture domain knowledge) to guide theorem proving search. There are fundamental problems in using "sophisticated models" within a resolution based system. The work of Sandford [Sandford,1980] provides a way for using "sophisticated models" to guide resolution search in a manner that preserves soundness and completeness. But as is well known, intuitively good models often do not have the right kinds of effects in reducing the resolution search space. Our current work is directed towards identifying and controlling some of the underlying causes of this phenomenon.

Results:

We have implemented an experimental system that uses the principles of the MDS meta-theory to learn from examples of situations that are presented to it. We have used this system to develop characterizing descriptions of letters in the English alphabet, based on examples of letters that are presented to it. We have also used the system to identify relational identities of the kind "father.father = grandfather" from examples of databases of family relationships

We are currently implementing a system for "reasoning with residues" with the objective of automatically constructing procedures that solve certain kinds of "goal satisfaction problems" in a domain using a given first order domain theory.

References:

Srinivasan:

[1973] Architecture of Coherent Information Systems: A general problem solving system", IJCAI-3, pp.618-628. Republished in IEEE-TC., Vol. C-25, 4, pp.390-402 (1976).

[1973] Programming over a Knowledge Base, the basis for Automatic Programming, Department of Computer Science Report No. SOSAP-TR-5, Rutgers University, New Brunswick, N.J. 08903

[1977] The Meta-Description System: A system to generate Intelligent Information Systems, Part I: The Model Space", Department of Computer Science Report SOSAP-TR-20A.

[1980] Knowledge Based Learning Systems, Department of Computer Science Report, DCS-TR-89.

[1981] Knowledge Representation and Problem Solving in MDS, Department Computer Science Report, DCS-TR-90, Jan, 1981.

Sandford:

[1980] Using Sophisticated Models in Resolution Theorem Proving, "Lecture Notes in Computer Science", Vol. 90, Eds. Goos and Hartmanis, Springer-Verlag.

[VanLehn] A THEORY OF HOW PEOPLE LEARN PROCEDURES

Personnel: Kurt VanLehn, John Seely Brown and other members of the Cognitive and Instructional Sciences group at Xerox PARC.

Objectives, methods, and results:

A theory of how people learn procedures is being developed. The objective of the research discussed here is not only to implement a model of procedure learning and test it with data from people learning procedures, but to state and defend principles that constrain the model. These principles are the basis of the theory. Observational, descriptive and explanatory adequacy, three criteria that play important roles in evaluating linguistic theories, are used in defending the principles of the theory being developed.

The theory is being tested using data from students learning elementary mathematical procedures, such as multi-digit subtraction. The advantage of studying such procedures is that they are virtually meaningless for most students. Observations in our lab have confirmed what many educators bemoan: most students neither use nor understand the underlying semantic or teleologic basis of arithmetic procedures which are rooted in the base-ten, place-value notation used for numbers. For initial studies in procedure learning, such semantic-free understanding is ideal for the same reason that nonsense words were useful in early psychological experiments. The meaninglessness of the procedures assures that simplicity in their formal representation is well-founded, allowing the theory to concentrate on learning.

When the procedural behavior of students who are in the midst of learning an arithmetic procedure is examined, about half of the students appear to be systematically following a procedure (but not the correct one since by assumption they have not learned it yet). John Seely Brown and Richard Burton discovered that in all cases these student's faulty procedures could be represented as a minor variations or bugs to the correct procedure (Brown & Burton, 1978). This particularly simple representation of behavior led to concentration on the learning of these individuals, namely those that exhibit their partial knowledge as buggy procedures.

A major result is that a certain kind of local problem solving appears to play a role in generating bugs (Brown & VanLehn, 1980; Brown & VanLehn, 1981). Briefly, when a stud[illegible] has unsuccessfully applied a procedure to a given problem, [illegible]e will attempt a repair. Let us suppose that he is missing a fragment of some correct procedural skill, such as multi-digit subtraction, either because he never learned the fragment or maybe he forgot it. Attempting to rigorously follow the impoverished procedure will

often lead to an impasse. That is a situation in which some current step of the procedure dictates a primitive action which the student believes cannot be carried out. For example, an impasse would follow from an attempt to decrement a zero during borrowing, provided the student knows (or discovers) that the decrement primitive has a precondition that its input argument can't be zero. When a constraint or precondition is violated, the student, unlike a typical computer program, is not apt to just quit. Instead he will often be inventive, invoking his problem solving skills in an attempt to repair the impasse so that he can continue to execute the procedure, albeit in a potentially erroneous way. We believe that many bugs can best be explained as patches derived from repairing a procedure that has encountered an impasse while solving a particular problem.

Other mechanisms than repair are necessary to account for the learning exhibited by "buggy" students. In particular, a model is needed for the acquisition of the incomplete or impoverished procedures that are the subject of repair. This is the problem currently under study.

One potential model for procedure learning that readily generates incomplete, partially learned procedures is an extension of the learning by example or "conjunctive generalization" work of Winston, Hayes-Roth, Michalski, Vere, Mitchell and others. It can be shown that part of a student's knowledge about a procedure is better represented by a certain kind of and-or graph (AOG) than by a less structured representation such as a production system (VanLehn, 1980). To extend a conjunctive generalization algorithm so that AOGs can be induced, one must deal with unconstrained disjunctions. In the work cited above, only one top level disjunction was allowed, thus allowing the algorithm to induce several alternative conjunctive hypotheses. But AOGs have disjunctions throughout their structure, not just at the top. Two algorithms have been found to induce AOGs, including a restricted form of AOGs that is isomorphic to the class of context-free grammars (VanLehn, forthcoming).

These algorithms shed light on the so-called "new terms" problem of induction. They invent new terms by looking for several occurrences of the same disjunction in the AOG. Naming the disjunction and substituting the resulting new term into the AOG simplifies its structure and sets the stage for collapsing nesting subtrees into loops. Forming loops corresponds to discovering recursive subprocedures when the AOG is viewed as a procedure.

Despite the success in extending conjunctive generalization to AOGs, it is too early to know whether these algorithms can be used in an adequate model of human procedure learning. Current research involves implementing a full-fledged procedure learner based on one of the AOG learning algorithms and interfacing it with the problem solver used to model repair. The empirical predictions of the resulting model can then be readily compared to the extensive data that has been collected over the last year on bugs in subtraction (VanLehn & Friend, 1980).

Support:

Grant N0014-78-C-0022 from the Office of Naval Research to the Learning Research and Development Center, University of Pittsburgh.

References:

Brown, J.S. & Burton, R.B. Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 1978, 2, 155-192.

Brown, J.S. & VanLehn, K. Towards a generative theory of bugs. In Addition and Subtraction: a developmental perspective. T. Carpenter, J. Moser, and T. Romberg (Eds.) Hillsdale, N.J.: Lawrence Erlbaum Associates, 1981.

Brown, J.S. & VanLehn, K. Repair Theory: A generative theory of bugs in procedural skills. Cognitive Science, 1980, 4(4).

VanLehn, K. On the representation of procedures in Repair Theory. Pittsburgh: University of Pittsburgh, Learning Research and Development Laboratory technical report, 1980.

VanLehn, K. Algorithms for learning by examples. Palo Alto, California: Xerox Palo Alto Research Centers technical report, forthcoming.

VanLehn, K. and Friend, J. Results from DEBUGGY: An analysis of systematic subtraction errors. Palo Alto California: Xerox Palo Alto Research Centers technical report, 1980.

[Whitehill] SELF-CORRECTING GENERALIZATION

Personnel: Stephen B. Whitehill, Artificial Intelligence Project, University of California at Irvine, Irvine, CA 92717.

Objectives, methods, and results:

For the past year, I have been studying the problem of incremental generalization from examples. In particular, the problem of recovery from initially misleading input sequences has been studied. This research has been conducted on my own, except for some helpful assistance from my advisor, Dennis Kibler. The work is not currently funded.

Here is a brief description of the research. A program has been written which accepts a series of input/output pairs and builds a production system. The production system is modified as new I/O pairs are seen. Specific instances are generalized into rules. This is known as incremental generalization. The program's rule insertion and generalization algorithms maintain a "minimal" set of rules.

The problem domain chosen was that of language morphology. That is, the generation of rules such as *->*ED from examples such as "jumped" and "walked". This was thought to be a good test domain because there are general rules such as *->*ED, exception rules such as *E->*ED (i.e. "used") and specific exceptions such as GO->WENT. Within this domain, the minimal rule set was informally defined to be those rules most likely to be found in a dictionary. The rule set found by the program on a complicated French morphology example was found to be the same as the rules found in a French-English dictionary.

The problem with incremental generalization is that if the program encounters unusual examples first it may over-generalize. The idea of associating positive and negative evidence with each generalization is useful if the program is to recover from initially misleading input sequences. By using this notion, new rules which are blocked by previous over-generalizations will eventually win out when there is sufficient evidence for them.

For example, when the system is given "churches", "buses" and "matches" the program assumes the rule is *->*ES. When there are a sufficient number of rules of the form *B->*BS, *C->*CS, etc. (evidence for the blocked rule *->*S) the system undoes the *->*ES generalization and replaces it with *CH->*CHES, *S->*SES, etc. Then the *->*S rule, no longer blocked by *->*ES, is put into the production system.

The important result of the work is this ability to recover from

misleading input. Since the choice of language morphology was arbitrary, any incremental learning system could improve its performance using these techniques.

Current research is centered on proving the assertion that our rule insertion algorithm maintains a minimal set of rules. A formal definition of the minimal rule set and a proof of the above assertion should appear shortly. Also, other problem domains are being considered for the purpose of demonstrating that the techniques really are problem-domain independent.

References

A more complete abstract of this work appears in the 1980 AAAI proceedings under the title, "Self-Correcting Generalization." A detailed description of the research appears in the U. C. Irvine Computer Science Technical Report of the same name (No. 149).

A BIBLIOGRAPHY ON MACHINE LEARNING

compiled by Bernard Nudel and Paul E. Utgoff

The following collection of papers is a bibliography for work relating to machine learning. The word "learning" is often used in the company of terms such as "induction","generalization","adaptation", "concept formation", "knowledge aquisition", etc. Buchanan et al.[72] and Smith et al.[349], characterize a "Learning System" as "any system which uses information obtained during one interaction with its environment to improve its performance during future interactions".

We can view the related terms above (induction, generalization, adaptation, etc) as often used, basic techniques for achieving the desired learning. In particular, induction can occur without learning (and vice versa), and induction work unrelated to learning has not been cited in the bibliography. Buchanan et al.[72] provide a useful framework in which to relate learning per se to the various techniques available.

In a wider context, learning can itself be viewed as a form of problem solving. Simon and Lea [344] emphasize this, and present a general framework for both (standard) "problem solving" and "rule induction". Amarel's "derivation problem" - "formation problem" spectrum [5] also is useful in viewing learning-related issues in the context of problem solving in general.

Apart from sources contributed by the authors, the bibliography owes much to personal bibliographies contributed by Saul Amarel, Dana Angluin, Ryszard Michalski, Tom Mitchell and Carl Smith, whom we greatfully thank. The contents of this bibliography reflect our own current view of what is significant. Although strongly influenced by our contributors, they are not accountable for our selections. Comments from the reader are welcomed. We exclude, or demphasize, areas where extensive bibliographies already exist, such as induction of numerical discriminants (pattern recognition), the adaptive system approach to learning (within control theory), and data interpolation and approximation by curve fitting (within numerical analysis).

We have placed the references below into classes. However, we were unfamiliar with a number of the references supplied by the other contributors. For this reason the assignment of references to classes is not complete and is not necessarily accurate in all cases. Again, comments are welcomed.

The categories of the taxonomy essentially structure the references according to the representation of what is induced in the process of learning. Although as mentioned, learning can occur without induction, this is one relatively clear cut and generally applicable criterion for classification. The particular category choices and their meanings appear in the headings and short descriptions, which preceed their respective cross-reference groups at the end of the bibliography. Some references deal with more than one of these categories while some papers defy classification this way, but nevertheless belong in the bibliography.

o 1. Adleman, L. and Blum, M. (1975), Inductive inference and unsolvability, technical report, Dept. Electrical Engineering and Computer Science and the Electronics Research Lab., U. Calif. at Berkeley.

p 2. Amarel, S. (1962), On the automatic formation of a computer program which represents a theory, in: Yovits, M., Jacobi, G. and Goldstein, G.(Eds.) *Self-organizing Systems*, pp. 107-175, Spartan Books, Washington, D.C.

3. Amarel, S. (1968), On representations of problems of reasoning about actions, in: Michie, D.(Ed.) *Machine Intelligence 3*, University of Edinburgh Press, Edinburgh.

p 4. Amarel, S. (1971), Representations and modeling in problems of program formation, in: Meltzer, B. and Michie, D.(Eds.) *Machine Intelligence 6*, University of Edinburgh Press, Edinburgh.

o 5. Amarel, S. (1978), Basic Themes and Problems in Current AI Research, CBM-TR-91, Dept. Computer Science., Rutgers U., New Brunswick, N.J.

oh 6. Anderson, J. R., Kline, P. J. and Beasely, C. M. (1977), A theory of the acquisition of cognitive skills, ONR Technical report 77-1, Yale University.

g 7. Anderson, J. R. (1977), Induction of Augmented Transition Networks, in: *Cognitive Science 1*.

oc 8. Anderson, J. R., Kline, P. J. and Beasley, C. M. (1979), A general learning theory and its application to schema abstraction, in: Bower(Ed.) *The Psychology of Learning and Motivation 13*.

oh 9. Anderson, J. R. (July 1980), A general learning theory and its application to the acquisition of proof skills in geometry, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 10. Andrews, A. M. (1959), Learning machines, in: *Proc. of the Symposium on the Mechanization of Thought Processes*, H.M. Stationary Office, London, England.

11. Angluin, D. (1978), On the complexity of minimum inference of regular sets, in: *Information and Control 39(3)*, pp. 337-350.

g 12. Angluin, D. (1979), Inductive inference of formal languages from positive data, technical report, Dept of Mathematics., U. Calif at Santa Barbara.

c 13. Angluin, D. (May 1979), Finding patterns common to a set of strings, in: *Eleventh Annual ACM Symposium on the Theory of Computing*, pp. 130-141.

p 14. Angluin, D. and Smith, C. H. (1981), A discrete theory of automatic program synthesis (In preparation).

h 15. Anzai, Y. and Simon, H. (1979), The theory of learning by doing, in: *Psychological Review 36(2)*, pp. 124-140.

h 16. Anzai, Y. (1978), Learning strategies by computer, in: *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 181-190.

h 17. Anzai, Y. (1978), How one learns strategies: Processes and representation of strategy acquisition, in: *Proceedings of the 3rd conference on AISB*, Hamburg, Gesellschaft fur Informatik.

18. Aubin, R. (August 1977), Strategies for mechanizing structural induction, in: *IJCAI5*, pp. 363-369, Cambridge, Mass.

Banerji, R. B. (1962), The description list of concepts, in: *J.A.C.M.*

Banerji, R. B. (1964), A language for description of concepts, in: *General Systems 9.*

Banerji, R. B. (1964), Computer programs for the generation of new concepts from old ones, in: Steinbuch, K. and Wagner, S.(Eds.) *Neure Ergebnisse der Kybernetik*, pp. 336, Oldenberg-Verlag, Munich.

Banerji, R. B. (1976), Learning to solve games and puzzles, in: Simon, J. C.(Ed.) *Computer Oriented Learning Processes*, Noordhoff, Leyden.

Banerji, R. B. (1976), A data structure which can learn simple programs from examples of input-output, in: Chen(Ed.) *Pattern Recognition and Artificial Intelligence*, Academic Press, New York.

Banerji, R. B. (1977), Learning in structural description languages, Temple University Report to NSF Grant MCS 76-0-200.

Banerji, R. B. (1979), Pattern recognition: structural description languages, in: Belzer, Holzman and Kent(Eds.) *Encyclopedia of Computer Science and Technology 12*, pp. 1, Marcel Dekker, New York.

Banerji, R. B. (1980), *Artificial Intelligence: A Theoretical Perspective*, Elsevier North Holland, Inc., New York.

Banerji, R. B. (July 1980), Requirements on and problems in description languages and induction, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

Banerji, R. B. and Mitchell, T. M. (June 1980), Description languages and learning algorithms: a paradigm for comparison, in: *Policy Analysis and Information Systems 4(2).*

Barnes, J. (1975), *Aristotle's Posterior Analytics*, Claredon Press, Oxford.

Barrow, H. G. and Popplestone, R. J. (1972), Relational descriptions in picture processing, in: Meltzer, B. and Michie, D.(Eds.) *Machine Intelligence 7*, pp. 377-396, American Elsevier, New York.

Barzdin, J. (1974), Two theorems on the limiting synthesis of functions, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 82-88, Latvian State U., Riga, U.S.S.R.

Barzdin, J. A. (1970), On decoding automata in the absence of an upper bound on the number of states, in: *Soviet Math Dokl*, pp. 1084-1051.

Barzdin, J. A. (1971), Prognostication of automata and functions, in: *Information Processings '71, Proceedings of the IFIP Congress 1*, pp. 81-84.

Barzdin, J. A. (1972), On synthesizing programs given by examples, in: *Lecture Notes in Computer Science 5*, Springer-Verlag.

Barzdin, J. A. and Freivald, R. V. (1972), On the prediction of general recursive functions, in: *Soviet Math Dokl 13*, pp. 1224-1228.

Barzdin, J. A. and Podnieks, K. M. (1973), The theory of inductive inference, in: *Proceedings of the Mathematical Foundations of Computer Science*, pp. 9-15 (Russian).

Barzdin, J. A. (1974), Uber eine eigenschaft limitar berech eubarer funktionale, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 20-24, Latvian State U., Riga, U.S.S.R.

Barzdin, J. A. and Freivald, R. V. (1974), Prediction and limiting synthesis of r.e. classes of functions, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 101-111, Latvian State U., Riga, U.S.S.R.

Barzdin, J. A., Kinber, E. B. and Podnieks, K. M. (1974), Concerning synthesis and prediction of functions, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 117-128, Latvian State U., Riga, U.S.S.R.

Biermann, A. W. (September 1975), A basis for the acquisition of procedures from protocols, in: *IJCAI4*, pp. 226-231, Cambridge, Mass.

Berger, J. and Pair, C. (1978), Inference for regular bilanguages, in: *J. Computer and System Sciences 15*, pp. 100-122.

Bernoulli, J. (1713), *Ars coniectandi*, Basel.

Bever, T. G., Fodor, J. A. and Garrett, M. (1968), A formal limitation of associationism, in: Dixon, T. R. and Horton, D. L.(Eds.) *Verbal Behavior and General Behavior Theory*, Prentice-Hall, Englewood Cliffs, N.J.

Bierman, A. W. and Feldman, J. A. (January 1971), A survey of results in grammatical inference, in: *Proceedings International Conference on Frontiers in Pattern Recognition*, Honolulu.

Bierman, A. W. and Feldman, J. A. (1972), A survey of results in grammatical inference, in: Watanabe, S.(Ed.) *Frontiers of Pattern Recognition*, pp. 31-54, Academic Press, New York.

Bierman, A. W. (1976), Regular LISP programs and their automatic synthesis from examples, technical report CS-1976-12, Duke University, North Carolina.

Bierman, A. W. (August 1978), The inference of regular LISP programs from examples, in: *IEEE Transcations on Systems, Man, and Cybernetics SMC-8(8)*, pp. 585-600.

Biermann, A. W. (1972), On the inference of turing machines from sample computations, in: *Artificial Intelligence 3*, pp. 181-198.

Biermann, A. W. and Feldman, J. A. (1972), On the synthesis of finite-state machines from samples of their behaviour, in: *IEEE Trans on Computers C-21*, pp. 592-597.

Biermann, A. W., Baum, R. I. and Petry, F. E. (1975), Speeding up the synthesis of programs from traces, in: *IEEE Trans on Computers C-24*, pp. 122-136.

Biermann, A. W. and Krishnaswamy, R. (1976), Constructing programs from example computations, in: *IEEE Trans on Software Engineering SE-2*, pp. 141-153.

Biermann, A. W. and Smith, D. R. (1977), The hierarchical synthesis of LISP scanning programs, in: B. Gilchrist(Ed.) *Information Processing 77*, pp. 41-45, North Holland, Amsterdam.

Blake, R. M., Ducasse, C. J. and Madden, E. H. (1960), *Theories of Scientific Method: The Renaissance through the Nineteenth Century*, U. Washington Press, Seattle.

Blum, L. and Blum, M. (1975), Toward a mathematical theory of inductive inference, in: *Information and Control 28*, pp. 125-155.

Blum, L. and Blum, M. (March 1973), Inductive Inference: A Recursive Theoretic Approach, Memo No. ERL-M386, ERL, U. California at Berkeley.

Book, W. F. (1908), The psychology of skill with special reference to its acquisition in typewriting, University of Montana, Missoula, Montana (Facsimile in The Psychology of Skill, Armo Press, New York, 1973).

Bradshaw, G. L., Langley, P. and Simon, H. A. (1980), BACON.4: The discovery of intrinsic properties, in: *Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 19-25.

Brown, D. J. H. (1977), Concept learning by feature value interval abstraction, in: *Proceedings of the Workshop on Pattern Directed Inference Systems, SIGART Newsletter 63*, pp. 55-60.

Brown, J. S. (1973), Steps toward automatic theory formation, in: *Proceedings of IJCAI3*, pp. 20-23, Stanford University.

Brown, J. S. (1975), Steps toward a theoretical foundation for complex, knowledge based CAI, BBN Report 3135, Cambridge, Mass. (authors are et al.).

Brown, J. S. and Burton, R. B. (1978), Diagnostic models for procedural bugs in basic mathematical skills, in: *Cognitive Science 2*, pp. 155-192.

h 62. Brown, J. S. and VanLehn, K. (1980), Repair theory: a generative theory of bugs in procedural s kills, in: *Cognitive Science 4(4).*

63. Brown, J. S. and VanLehn, K. (1981), Towards a generative theory of bugs, in: Carpenter, T., Moser, J. and Romberg, T.(Eds.) *Addition and Subtraction: a developmental perspective*, Lawrence Erlbaum Associates, Hillsdale, N. J.

o 64. Brown, M. F. and Tarnlund, S-A. (August 1977), Inductive reasoning in mathematics, in: *Proc. Fifth IJCAI*, M.I.T., Cambridge, Mass.

o 65. Bruner, J. S., Goodnow, J. J. and Austin, G. A. (1956), *A Study of Thinking*, Wiley, New York.

h 66. Buchanan, B. G., Feigenbaum, E. A. and Lederberg, J. (1971), A heuristic programming study of theory formation in sciences, in: *Proc of IJCAI2*, London.

hc 67. Buchanan, B. G., Feigenbaum, E. A. and Sridharan, N. S. (1972), Heuristic theory formation: data interpretation and rule formation, in: Meltzer, B. and Michie, D.(Eds.) *Machine Intelligence 7*, pp. 267-290, Halsted Press, Wiley, New York.

h 68. Buchanan, B. G. and Sridharan, N. S. (1973), Rule formation on non-homogeneous classes of objects, in: *Proc. Third IJCAI*, Stanford U., Stanford, Calif.

hc 69. Buchanan, B. G. (1976), Scientific theory formation by computer, in: Simon, J. C.(Ed.) *Computer Oriented Learning Processes*, Noordhoff, Leyden.

hc 70. Buchanan, B. G. and Mitchell, T. M. (1978), Model-directed learning of production rules, in: Waterman, D. A. and Hayes-Roth, F.(Eds.) *Pattern-Directed Inference Systems*, Academic Press, New York.

h 71. Buchanan, B. G. and Feigenbaum, E. A. (1978), DENDRAL and META-DENDRAL: their applications dimension, in: *Artificial Intelligence 11*, pp. 5-24, North-Holland.

o 72. Buchanan, B. G., Mitchell, T. M., Smith, R. G. and Johnson, C. R. Jr. (1978), Models of learning systems, in: *Encyclopedia of Computer Science and Technology 11*, Dekker (also Stanford report STAN-CS-79-692).

73. Burge, J. and Hayes-Roth, F. (1976), A novel pattern learning and recognition procedure applied to the learning of vowels, Technical Report, Carnegie-Mellon University, Pittsburgh, PA.

p 74. Burstall, R. M. and Darlington, J. (1977), A transformation system for developing recursive programs, in: *Journal of the ACM 24(1)*, pp. 44-67.

75. Carbonell, J. G. (1979), Towards a self-extending parser, in: *Proceedings of the 17th Meeting of the Association for Computational Linguistics*, pp. 3-7.

76. Carbonell, J. G. (1980), Metaphor - A key to extensible semantic analysis, in: *Proceedings of the 18th Meeting of the Association for Computational Linguistics.*

77. Carbonell, J. G. (August 1980), DELTA-MIN: a search-control method for information-gathering problems, in: *AAAI1*, Stanford, CA.

c 78. Carbonell, J. G. (July 1980), Interactive concept acquisition on hierarchical memory structures, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 79. Carnap, R. (1952), *The Continuum of Inductive Methods*, The U. of Chicago Press, Illinois.

o 80. Carnap, R. (1962), The aim of inductive logic, in: Nagel, E., Suppes, P. and Tarski, A.(Eds.) *Logic, Methodology and Philosophy of Science*, pp. 303-318, Stanford University Press, Stanford, California.

o 81. Carnap, R. (1963), Variety, analogy and periodicity in inductive logic, in: *Philosophy of Science 30*, pp. 222-227.

o 82. Carnap, R. and Jeffrey, R. (1971), *Studies in Inductive Logic and Probability*, U. of Calif Press, Berkeley, Calif.

o 83. Case, J. and Smith, C. (1981), Comparison of identification criteria for mechanized inductive inference, TR-154, Dept. Computer Science., State U. of New York at Buffalo (to be published).

84. Case, J. and NgoManguelle, S. (1981), Refinements of inductive inference by poperian machines (to appear in Kybernetika).

85. Case, J. and Smith, C. (1978), Anomaly hierarchies of mechanized inductive inference, in: *Proceedings of the 10-th Symposium on the Theory of Computing*, pp. 314-319 (is an preliminary version of Case and Smith 1981, below).

r 86. Chen, C. H. (1977), Statistical pattern recognition- review and outlook, in: *IEEE Systems, Man and Cybernetics Newsletter 6(4)*, pp. 7-8.

87. Chisolm, I. H. and Sleeman, D. H. (1979), An aide for theory formation, in: Michie, D.(Ed.) *Expert Systems in the Micro-Electronic Age*, pp. 202-212, Edinburgh University Press.

o 88. Churchman, C. W. (1948), *Theory of Experimental Inference*, Macmillan, New York.

o 89. Churchman, C. W. and Buchanan, B. G. (1969), On the design of inductive systems: some philosophical problems, in: *Brit. J. Phil. Sci. 20.*

90. Clancey, W. (January 1979), Tutoring rules for guiding a case model dialogue, in: Brown and Sleeman, D.(Eds.) *International Journal of Man-Machine Studies.*

91. Clancey, W. J., Shortliffe, E. H. and Buchanan, B. G. (October 1979), Intelligent computer-aided instruction for medical diagnosis, in: *Proceedings of the Third Annual Symposium on Computer Applications in Medical Care*, Silver Spring, Maryland.

c 92. Cohen, B. L. (December 1977), A powerful and efficient structural pattern recognition system, in: *Artificial Intelligence 9(3).*

c 93. Cohen, B. L. (1978), CONFUCIUS, a structural pattern recognition and learning system, in: *Proceedings of the International Conference on Cybernetics and Society*, pp. 1443, Tokyo-Kyoto.

c 94. Cohen, B. L. (1978), A theory of structural concept formation and pattern recognition, Ph.D. thesis, Department of Computer Science, University of South Wales.

c 95. Cohen, B. L. (1978), A powerful and efficient structural pattern recognition system, in: *Artificial Intelligence* 9, pp. 223.

c 96. Cohen, B. L. and Sammut, C. A. (1978), Pattern recognition and learning with a structural description language, in: *Proceedings of the IJCPR4*, pp. 394, Kyoto, Japan.

97. Cohen, D. (1980), Knowledge Based Theorem Proving and Learning, Ph.D. Thesis, Technical report CMU-CS-80-115, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA.

g 98. Cook, C. M. and Rosenfeld, A. (1976), Some experiments in grammatical inference, in: Simon, J. C.(Ed.) *Computer Oriented Learning Processes*, Noordhoff, Leyden.

g 99. Cook, C. M., Rosenfeld, A. and Aronson, A. R. (1976), Grammatical inference by hill-climbing, in: *Information Sciences 10*, pp. 59-80.

o 100. Coulon, D. and Kayser, D. (1978), Learning Criterion and Inductive Behavior, in: *Pattern Recognition 10(1)*, pp. 19-25.

g 101. Coulon, D. and Kayser, D. (1979), Construction of natural-language sentence acceptors by a supervised learning technique, in: *IEEE Trans on Pattern Analysis and Machine Intelligence PAMI-1*, pp. 94-99.

g 102. Crespi-Reghizzi, S. (1971), Reduction of enumeration in grammar aquisition, in: *Proc. Second IJCAI*, British Computer Society, London.

g 103 Crespi-Reghizzi, S. (1972), An effective model for grammar inference, in: *Information Processings 71*, pp. 524-529, North-Holland.

g 104 Crespi-Reghizzi, S., Melkanoff, M. A. and Lichten, L. (February 1973), The use of grammatical inference for designing programming languages, in: *Communications of the ACM 16(2)*, pp. 83-90.

105 Daley, R. (1977), On the inference of optional descriptions, in: *Theoretical Computer Science 4*, pp. 301-319.

g 106 Derwing, B. L. (1973), *Transformational Grammar as a Theory of Language Aquisition*, Cambridge U. Press, Cambridge, England.

r 107 Diday, E. and Simon, J. C. (1976), Clustering Analysis, in: Fu, K. S.(Ed.) *Communication and Cybernetics 10*, Springer-Verlag, Berlin, Heidelberg, New York.

c 108. Dietterich, T. (October 1978), Description of Inductive Program INDUCE 1.1, Internal Report, Department of Computer Science, University of Illinois, Urbana-Champaign.

c 109 Dietterich, T. G. and Michalski, R. S. (August 1979), Learning and generalization of characteristic descriptions: evaluation criteria and comparative review of selected methods, in: *Proceedings of IJCAI6*, pp. 223-231, Tokyo, Japan.

c 110. Dietterich, T. G. (October 1979), The Methodology of Knowledge Layers for Inducing Descriptions of Sequentially Ordered Events, M.S. Thesis, Department of Computer Science, University of Illinois, Urbana.

c 111. Dietterich, T. G. (August 1980), Applying general induction methods to the card game Eleusis, in: *AAAI-80*, pp. 218-220, Stanford University.

c 112. Dietterich, T. G. (July 1980), Multiple-model induction in Eleusis, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

g 113. Doucet, P. G. (1974), The syntactic inference problem for DOL-sequences, in: *L-Systems., Lecture Notes in Computer Science 15*, pp. 146-161, Springer-Verlag.

114. Doyle, J. (1979), A truth maintainance system, in: *Artificial Intelligence 12(3)*.

ec 115. Duda, R. O. and Hart, P. E. (1973), *Pattern Classification and Scene Analysis*, Wiley, New York.

h 116. Elcock, E. W. and Murray, A. M. (1967), Experiments with a learning component in a go-muku playing program, in: Collins and Michie, D.(Eds.) *Machine Intelligence 1*, pp. 87-103, Oliver & Boyd, London.

c 117. Ernst, G. W. and Sherman, R. (1969), Recognizing concepts in terms of other concepts, in: *Pattern Recognition 2*, pp. 301.

h 118. Ernst, G. W. (January 1974), Mechanical discovery of certain heuristics, Report 113, Jennings Computer Center, Case Western Reserve University.

p 119. Ernst, G. W. and Hookway, R. J. (1975), Formulating inductive assertions for program verifications, technical report, Case Western Reserve University, Cleveland, Ohio.

g 120. Feigenbaum, E. A. (1963), The simulation of verbal learning behaviour, in: Feigenbaum, E. A. and Feldman, J.(Eds.) *Computers and Thought*, pp. 297-309, McGraw-Hill, New York (originally in Proc. Western Joint Computer Conf., 1961).

121. Feldman, J. (1963), Simulation behaviour in the binary choice experiment, in: Feigenbaum and Feldman(Eds.) *Computers and Thought*, McGraw-Hill, New York.

g 122 Feldman, J. (1972), Some decidability results on grammatical inference of best programs, in: *Math Systems Theory 10*, pp. 244-262.

r 123 Feldman, J. and Shields, P. (1977), Total complexity and the inference of best programs, in: *Math Systems Theory 10*, pp. 181-191.

g 124. Feldman, J. A. (1967), First Thoughts on Grammatical Inference, Artificial Intelligence Memo No. 55, Stanford U.

g 125. Feldman, J. A., Gips, J., Horning, J. J. and Reder, S. (1969), Grammatical complexity and inference, CS 125, Computer Science Deptartment, Stanford University.

g 126. Feldman, J. A. (1972), Some decidability results on grammatical inference and complexity, in: *Information and Control 20*, pp. 244-262.

g 127. Feliciangeli, H. and Herman, G. T. (1973), Algorithms for producing grammars from sample derivations: A common problem of formal language theory and developmental biology, in: *Journal of Computer and System Sciences 7*, pp. 97-118.

hc 128. Fikes, R. E., Hart, P. E. and Nilsson, N. J. (1972), Learning and executing generalized robot plans, in: *Artificial Intelligence 3*, pp. 251-288.

o 129. Findler, N. V. and McKinsie, W. R. (1969), Computer simulation of a self-preserving and learning organism, in: *Bulletin Math. Biophysics 31*, pp. 247-253.

h 130. Findler, N. V. (1977), Studies in machine cognition using the game of poker, in: *CACM 20(4)*, pp. 230-245.

o 131. de Finetti, B. (1972), *Probability, Induction, and Statistics -- The Art of Guessing*, John Wiley and Sons, New York.

c 132. Fox, M. S. and Hayes-Roth, F. (1976), Approximation techniques for the learning of sequential symbolic patterns, in: *Proceedings of the Third IJCPR*, pp. 616-620, Coronado, CA.

133. Fox, M. S. (1978), Knowledge structuring: An overview, in: *Proceedings of the 2nd Conference of the Canadian Society for Computational Studies of Intelligence*, Toronto, Ontario.

134. Fox, M. S. (approx 1980), Knowledge Structuring: Knowledge Accomodation and Discovery Using Specialization and Planning, Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA. (to appear).

135. Fox, M. S. (July 1980), Reasoning with incomplete knowledge in a resource-limited environment: integrating reasoning and knowledge acquisition, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

136. Fredkin, E. (1964), Techniques using LISP for automatically discovering interesting relations in data, in: Berkely and Bobrow(Eds.) *The Programming Language LISP*, Information International, Cambridge, Mass.

137. Freivald, R. (1975), On complexity and optimality of the computation in the limit, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 155-173, Latvian State U., Riga, U.S.S.R.

p 138. Freivald, R. V. (1974), On the limit synthesis of numbers of general recursive functions in various computable numerations, in: *Soviet Math Dokl 15*, pp. 1681-1683.

139. Freivald, R. V. (1974), Uniform and non uniform predictability, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 89-100, Latvian State U., Riga, U.S.S.R.

140. Freivald, R. V. and Kinber, E. B. (1977), Identification in the limit of minimal Godel numbers, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 1-34, Latvian State U., Riga, U.S.S.R. (in Russian).

141. Freivald, R. V. and Wiehagen, R. (1979), Inductive inference with additional information, in: *Elektronische Infomationsverarbeitung und Kybernetik 15(4)*, pp. 179-185.

o 142. Freudenthal, H. (1961), The Concept and the Role of the Model in Mathematics and Natural and Social Sciences, in: Hintikka, J.(Ed.) *Synthese Library*, Reidel Pub. Co.

143. Friedberg, R. (1958), A learning machine: part 1, in: *IBM J. of Research and Development 2(1)*, pp. 1-13.

144. Friedberg, R., Dunham, B. and North, T. (1959), A learning machine: part 2, in: *IBM J. of Research and Development 3(3)*, pp. 282-287.

145. Friedberg, R. M. (1958), A learning machine: part i, in: *IBM Journal 2*, pp. 2-13.

g 146. Fu, K. S. (1977), *Syntactic Pattern Recognition, Applications*, Springer-Verlag, New York.

rg 147. Fu, K. S. (1979), Pattern Recognition: discriminant and syntactic methods, in: *Encyclopedia of Computer Science and Technology 12*, Marcel Dekkar, New York.

r 148. Fu, K. S. (1968), *Sequential Methods in Pattern Recognition and Machine Learning*, Academic Press, New York.

g 149. Fu, K. S. and Booth, T. L. (1975), Grammatical inference: introduction and survey-part I, in: *IEEE Transactions on SMC 5(1)*, pp. 95-111.

g 150. Fu, K. S. and Booth, T. L. (1975), Grammatical inference: introduction and survey-part II, in: *IEEE Transactions on SMC 5(4)*, pp. 409-423.

r 151. Fukanaga, K. (1972), *Introduction to Statistical Pattern Recognition*, Academic Press.

152. Gaines, B. R. (1976), Behaviour/structure transformations under uncertainty, in: *Int. J. of Man-Machine Studies 8*, pp. 337-365.

g 153. Gaines, B. R. (1979), Maryanski's grammatical inferencer, in: *IEEE Trans on Computers C-28*, pp. 62-64.

154. Gelernter, H. L., Sanders, A. F., Larsen, D. L., Agerwal, K. K., Boivie, R. H., Spritzer, G. A. and Searleman, J. E. (September 1977), Empirical explorations of SYNCHEM, in: *Science 197(4308)*, pp. 1041-1049.

155. German, S. M. and Wegbreit, B. (March 1975), A synthesizer of inductive assertions, in: *IEEE Transactions on Software Engineering SE-1(1)*, pp. 68-75.

g 156. Gill, A. (1961), State-identification experiments in finite automata, in: *Information and Control 4*, pp. 132-154.

co 157. Glanc, A. (1973), Theory formation by machine: a general framework of the GOLEM system, in: *IEEE Trans. on Systems, Man, and Cybernetics*, pp. 342-343.

p 158. Goetze, B. and Klette, R. (1974), Some properties of limit recursive functions, in: *Lecture Notes in Computer Science 28*, pp. 88-90, Springer-Verlag.

p 159. Gold, E. M. (1965), Limiting recursion, in: *J. Symbolic Logic 30*, pp. 28-48.

g 160. Gold, E. M. (1967), Language identification in the limit, in: *Information and Control 10*, pp. 447-474.

161. Gold, E. M. (1972), System identification via state characterization, in: *Automatica 8*, pp. 621-636.

g 162. Gold, E. M. (1978), Complexity of automaton identification from given data, in: *Information and Control 37*, pp. 302-320.

h 163. Goldstein, I. P. and Grissom, E. (1977), Annotated Production Systems: a model for skill acquisition, in: *IJCAI5*, pp. 311-317, Cambridge, Mass.

h 164. Goldstein, M. M. (1977), The mechanical discovery of problem solving strategies, Report ESCI-77-1, Case Institute of Technology, Case Western Reserve U.

g 165. Gonzalez, R. C. and Thomason, M. G. (1978), *Syntactic Pattern Recognition, An Introduction*, Addison-Wesley, Reading, Mass.

o 166. Good, I. (1971), The probabilistic explication of information, evidence, suprise, causality, explanation, and utility, in: Godambe and Sprott(Eds.) *Foundations of statistical inference*, pp. 108-122, Holt, Rinehart and Winston of Canada, Toronto.

ho 167. Griffith, A. K. (1974), A comparison and evaluation of three machine learning procedures as applied to the game of checkers, in: *Artificial Intelligence 5*, pp. 137-148.

p 168. Guiho, G. and Jouannaud, J. P. (1978), Program synthesis for a simple class of non-loop functions, 6, Laboratoire de Recherche en Informatique., Universite de Paris-Sud, Orsay.

169. Haas, N. and Hendrix, G. G. (August 1980), An approach to applying and acquiring knowledge, in: *AAAI1*, pp. 235-239, Stanford, CA.

o 170. Hacking, I. (1965), *Logic of Statistical Inference*, Harvard University Press, Cambridge, Mass.

o 171. Hajek, P. and Havranek, T. (1978), *Mechanizing Hypothesis Formation: Mathematical Foundations for a General Theory*, Springer-Verlag.

o 172. Hajek, P. (1975), On Logics of Discovery, in: *Lecture Notes in Computer Science 32*, pp. 30-45, Springer-Verlag.

p 173. Hardy, S. (September 1975), Synthesis of LISP functions from examples, in: *IJCAI4*, pp. 240-245, Cambridge, Mass.

g 174. Harris, L. R. (1977), A system for primitive natural language acquisition, in: *International Journal Man-Machine Studies*, pp. 153-206.

c 175. Hayes-Roth, F. (1974), Schematic classification problems and their solution, in: *Pattern Recognition 6*, pp. 105-113.

c 176. Hayes-Roth, F. and Mostow, D. (September 1975), An automatically compilable recognition network for structured patterns, in: *IJCAI4*, pp. 356-362, Cambridge, Mass.

c 177. Hayes-Roth, F. (May 1976), Collected papers on the learning and recognition of structured patterns, technical report, Carnegie-Mellon Department of Computer Science, Pittsburgh, PA.

c 178. Hayes-Roth, F. and Burge, J. (1976), Characterizing syllables as sequences of machine-generated labelled segments of connected speech: a study in symbolic pattern learning using a conjunctive feature learning and classification system, in: *Third IJCPR*, pp. 431-436, Coronado, CA.

c 179. Hayes-Roth, F. (1976), Representation of structured events and efficient procedures for their recognition, in: *Pattern Recognition 8*, pp. 141.

c 180. Hayes-Roth, F. (1976), Patterns of induction and associated knowledge acquisition algorithms, in: Chen, C.(Ed.) *Pattern Recognition and Artificial Intelligence*, Academic Press, New York.

c 181. Hayes-Roth, F. and McDermott, J. (August 1977), Knowledge acquisition from structural descriptions, in: *IJCAI5*, pp. 356-362, Cambridge, Mass.

hc 182. Hayes-Roth, F. (February 1977), Uniform representations of structured patterns and an algorithm for the induction of contingency-response rules, in: *Information and Control 33*, pp. 87-116.

c 183. Hayes-Roth, F. (1978), The role of partial and best matches in knowledge systems, in: Waterman, D. A. and Hayes-Roth, F.(Eds.) *Pattern-Directed Inference Systems*, pp. 557-576, Academic Press, New York.

c 184. Hayes-Roth, F., Klahr, P., Burge, J. and Mostow, D. J. (1978), Machine methods for acquiring, learning, and applying knowledge, Technical report R-6241, The RAND Corporation, Santa Monica, CA.

c 185. Hayes-Roth, F. and McDermott, J. (1978), An Interference Matching Techniques for Inducing Abstractions, in: *Communications of the ACM 21(5)*, pp. 401-410.

h 186. Hayes-Roth, F., Klahr, P. and Mostow, D. J. (July 1980), Advice-Taking and knowledge refinement: An Iterative View of Skill Acquisition, RAND Paper Series P-6517, The RAND Corporation, Santa Monica, CA.

h 187. Hayes-Roth, F., Klahr, P. and Mostow, D. J. (May 1980), Knowledge acquisition, knowledge programming, and knowledge refinement, R-2540-NSF, The Rand Corporation, Santa Monica, CA.

188. Hayes-Roth, F. (July 1980), Theory-driven learning: proofs and refutations as a basis for concept discovery, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

h 189. Hedrick, C. L. (July 1974), A Computer Program to Learn Production Systems Using a Semantic Net, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

h 190. Hedrick, C. L. (1976), Learning production systems from examples, in: *Artificial Intelligence 7(1)*, pp. 21-49.

o 191. Hendel, R. J. (1979), Mathematical learning theory: a formaized, axiomatic, abstract approach, in: *Information and Control 41*, pp. 67-117.

g 192. Herman, G. T. and Walker, A. D. (1972), The syntactic inference problem applied to biological systems, in: Meltzer and Michie(Eds.)*Machine Intelligence 7*, pp. 341-356, Edinburgh U. Press.

c 193. Herman, M. (July 1980), Towards learning descriptions of three-dimensional objects, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 194. Hintikka, J. (1965), Towards a theory of inductive generalization, in: *Proc 1964 Congress for Logic, Methodology and Philosophy of Science*, pp. 274-288, North Holland, Amsterdam.

o 195. Hintikka, J. (1965), On a combined system of inductive logic, in: *Acta Philosophica Fennica 18*, pp. 21-30.

o 196. Hintikka, J. and Hilpinen, R. (1966), Knowledge acceptance and inductive logic, in: Hintikka, J. and Suppes, P.(Eds.)*Aspects of Inductive Logic*, pp. 1-20, North-Holland, Amsterdam.

c 197. Holland, J. H. (September 1980), Adaptive algorithms for discovering and using general patterns in growing knowledge bases, in: *Policy Analysis and Information Systems 4(3)*.

o 198. Hormann, A. M. (1962), Programs for machine learning, Part 1, in: *Information and Control 5*, pp. 347-367.

o 199. Hormann, A. M. (1964), Programs for machine learning, Part 2, in: *Information and Control 7(1)*, pp. 55-77.

g 200. Horning, J. J. (August 1969), A Study of Grammatical Inference, Technical report CS-139, Ph.D. thesis, Department of Computer Science, Stanford University.

g 201. Horning, J. J. (1972), A procedure for grammatical inference, in: *Information Processing 71*, pp. 519-523.

c 202. Hunt, E. B. (1962), *Concept Learning: An Information Processing Problem*, Wiley, New York.

c 203. Hunt, E. B. and Hovland, C. I. (1963), Programming a model of human concept formation, in: Feigenbaum, E. A. and Feldman, J.(Eds.)*Computers and Thought*, pp. 310-325, McGraw-Hill, New York.

gc 204. Hunt, E. B. (1965), Selection and reception conditions in grammar and concept learning, in: *J. Verbal Learning Verbal Behavior 4*, pp. 211-215.

c 205. Hunt, E. B., Marin, J. and Stone, P. T. (1966), *Experiments in Induction*, Academic Press, New York.

c 206. Iba, G. A. (1979), Learning disjunctive concepts from examples, MIT AI memo 548, Cambridge, Mass.

207. Joshi, A. K. (1978), Some extensions of a system for inference on partial information, in: Waterman, D. A. and Hayes-Roth, F.(Eds.)*Pattern-Directed Inference Systems*, Academic Press, New York.

p 208. Jouannaud, J. P., Guiho, G. and Treuil, T. P. (1977), SISP/1 an interactive system able to synthesize functions from examples, in: *5-th International Joint Conference on Artificial Intelligence*, pp. 412-418.

p 209. Jouannaud, J. P. and Kodratoff, Y. (1979), Characterization of a class of functions synthesized from examples by a Summers-like method using a B.M.W. matching technique, in: *IJCAI6*, pp. 440-447, Tokyo, Japan.

p 210. Jouannaud, J. P. and Guiho, G. (1979), Inference of functions with an interactive system, in: Hayes, J. E., Michie, D. and Mikulich, L. I.(Eds.)*Machine Intelligence 9*, Ellis Horwood, Chichester.

r 211. Kanal, L. (1974), Patterns in Pattern Recognition: 1968-1974, in: *IEEE Trans on Information Theory IT-20(6)*, pp. 697-722.

r 212. Kanal, L. (1977), Current status, problems and prospects of pattern recognition, in: *IEEE Systems, Man and Cybernetics Newsletter 6(4)*, pp. 9-11.

r 213. Kanal, L. N. (1979), Problem-solving models and search strategies for pattern recognition, in: *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-1(2)*, pp. 193-201.

o 214. Kemeny, J. G. (1953), The use of simplicity in induction, in: *Philosophical Review 62*, pp. 391-408.

215. Kilburn, T., Grimsdale, R. L. and Sumner, F. H. (1959), Experiments in machine learning and thinking, in: *Information Processing of ICIP*.

216. Kinber, E. B. (1972), Frequency calculations of general recursive predicates and frequency enumeration of sets, in: *Soviet Math Dokl 13(4)*, pp. 873-876.

p 217. Kinber, E. B. (1977), On identification in the limit of minimal numbers for functions of effectively enumerable classes, in: Barzdin(Ed.)*Theory of Algorithms and Programs*, pp. 35-56, Latvian State U., Riga, U.S.S.R.

o 218. Kinber, E. B. (1977), On a theory of inductive inference, in: *Lecture Notes in Computer Science 56*, pp. 435-440, Springer-Verlag.

p 219. Klaus, P. J. (1976), Zur kompliziertheit der konsistenten erkennung von klassen allgemein-rekursiver funktionen, in: *Int. Symp. Algorithmische Kompliziertheit, lern, und Erkennungs-prozesse*, pp. 45-52.

g 220. Klein, S. and Kuppon, M. A. (1979), An Interactive Heuristic Program for Learning Transformational Grammar, Computer Science Dept., U. Wisconsin.

p 221. Klette, R. (1975), Unterklassen in der familie NUM aller effektiv numerierbaren mengen von eistelligen allgemein rekursiven funktionen, in: *Kompliziertheit von Lern und Erkennungs prozessen I*, pp. 62-66.

p 222. Klette, R. and Goetze, B. (1975), Limes-rekursive funktionen, in: *Elektronicshe Informationsverarbeitung und Kybernetik 11*, pp. 586-589 (in German).

p 223. Klette, R. (1976), Recognizing algorithms for recursive functions, in: *Elektronicshe Informationsverarbeitung und Kybernetik 12*, pp. 227-243 (in German).

p 224. Klette, R. (1977), A few results on the complexity of classes of identifiable recursive function sets, 4 (M) Sekt, Jena.

g 225. Knobe, B. and Knobe, K. (1976), A method for inferring context-free grammars, in: *Information and Control 31*, pp. 129-146.

226. Kochen, M. (1960), Experimental study of 'hypothesis formation' by computer, Research Report RC 294, IBM, Yorktown Heights, New York.

227. Kochen, M. (1961), An Experimental Program for the selection of disjunctive hypotheses, in: *Proc. Western Joint Computer Conference*, pp. 571-578.

c 228. Kochen, M. (1961), Experimental study of 'hypothesis-formation' by computer, in: Cherry, C.(Ed.)*4th London Symposium on Imformation Theory*, Butterworth, London and Washington, D.C.

229. Kochen, M. (1963), Some mechanisms in hypothesis selection, in: *Proc. Symp. on Mathematical Theory of Automata, N.Y. (1962)*, Polytechnic Press of the Polytechnic Institute of Brooklyn, N.Y.

230. Kochen, M. (1969), Experimental study of hypothesis formation by computer, in: *Proc. 1960 London Symp. on Information Theory*.

o 231. Kochen, M. (1971), Cognitive learning processes: an explication, in: Findler and Meltzer(Eds.)*Artificial Intelligence and Heuristic Programming*, Edinburgh U. Press.

232. Kochen, M. (1975), An algorithm for forming hypotheses about simple functions, in: *Third Milwaukee Symposium on Automatic Computation and Control*, Milwaukee, Wisconsin.

p 233. Kodaratoff, Y. (1979), A class of functions synthesised from a finite number of examples and a LISP program scheme, in: *Int. J. Computer and Information Sciences 8(6)*, pp. 489-521.

rh 234. Koffman, E. (March 1968), Learning through pattern recognition applied to a class of games, in: *IEEE Transactions on Systems Science and Cybernetics SC-4.*

hr 235. Koffman, E. B. (1968), Learning games through pattern recognition, in: *IEEE Trans on Systems Science and Cybernetics SSC-4(1).*

o 236. Kramer (1962), A note on the self-consistency definitions of generalization and inductive inference, in: *JACM*, pp. 280-281.

o 237. Kugel, P. (1977), Induction pure and simple, in: *Information and Control 35*, pp. 276-336.

o 238. Kuhn, T. S. (1970), *The Structure of Scientific Revolutions*, University of Chicago Press, Chicago (2nd edition).

o 239. Kyburg, H. (1964), Recent work in inductive logic, in: *American Philosophical Quarterly 1*, pp. 249-287.

o 240. Lakatos, I. (1976), *Proofs and refutations: The logic of mathematical discovery*, Cambridge University Press, Cambridge.

r 241. Langley, P. (1978), BACON.1: A general discovery system, in: *Proceedings of the Second National Conference of the Canadian Society for Computational Studies in Intelligence*, pp. 173-180.

o 242. Langley, P., Neches, R., Neves, D. and Anzai, Y. (1979), A Framework for learning procedures, Technical report, Carnegie-Mellon University, Pittsburgh, PA.

r 243. Langley, P. (1979), Rediscovering physics with BACON.3, in: *IJCAI6*, pp. 505-507, Tokyo.

r 244. Langley, P. W. (August 1977), BACON: A production system that discovers empirical laws, in: *IJCAI5*, pp. 344-346, Cambridge, Mass.

h 245. Langley, P. W., Neches, R., Neves, D. and Anzai, Y. (June 1980), A domain-independent framework for procedure learning, in: *Journal of Policy Analysis and Information Systems 4(2)*, pp. 163-197.

c 246. Larson, J. and Michalski, R. S. (1975), AQVAL/1 (AQ7) User's Guide and Program Description, report 731, Dept. Computer Science., U. Illinois, Urbana.

c 247. Larson, J. and Michalski, R. S. (1977), Inductive inference of VL decision rules, in: *Proceedings of the Workshop on Pattern Directed Inference Systems, SIGART Newsletter 63.*

c 248. Larson, J. (May 1977), Inductive inference in the variable-valued predicate logic system VL21: methodology and computer implementation, Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana, Illinois.

c 249. Larson, J. (May 1977), INDUCE-1: An Interactive Inductive Inference Program in VL21 Logic System, Report UIUCDCS-R-77-876, Department of Computer Science, University of Illinois, Urbana, Illinois.

250. Lenat, D. B. (1976), AM: an artificial intelligence approach to discovery in mathematics as heuristic search, Ph.D. thesis, Stanford University, Stanford, California.

251. Lenat, D. B. (1977), Automated theory formation in mathematics, in: *IJCAI5*, pp. 833-842, Cambridge, Mass.

o 252. Lenat, D. B. (December 1977), The ubiquity of discovery, in: *Artificial Intelligence 9(3)*, pp. 257-285.

op 253. Lenat, D. B., Hayes-Roth, F. and Klahr, P. (June 1979), Cognitive economy, RAND technical report N-1185-NSF, The RAND Corporation, Santa Monica, CA.

o 254. Lenat, D. B. (1980), The nature of heuristics, HPP-80-26, Heuristic Programming Project., Computer Science Dept., Stanford U., Stanford, Ca.

o 255. Levi, I. (1965), Deductive cogency in inductive inference, in: *Journal of Philosophy 62*, pp. 68-77.

o 256. Levi, I. (1967), *Gambling with Truth: An Essay on Induction and the Aims of Science*, Alfred A. Knopf, Inc., New York.

h 257. Lewis, C. H. (1978), Production system models of practice effects, Unpublished Ph.D. dissertation, University of Michigan.

o 258. MacKay, D. M. (1955), The epistemological problem for automata, in: Shannan and McCarthy(Eds.) *Automata Studies*, Princeton U. Press, Princeton, N.J.

g 259. Maryanski, F. J. and Booth, T. L. (1977), Inference of finite-state probabilistic grammars, in: *IEEE Trans on Computers C-26*, pp. 521-536 (See also Gaines 1979).

h 260. McDermott, J. (December 1978), ANA: An Assimilating and Accomodating Production System, Technical Report CMU-CS-78-156, Carnegi-Mellon University, Pittsburgh, PA.

261. McDermott, J. (1979), Learning to use analogies, in: *Proc. Sixth IJCAI, Tokyo.*

co 262. Meltzer, B. (1970), The semantics of induction and the possibility of complete systems of inductive inference, in: *Artificial Intelligence 1*, pp. 189-192.

c 263. Michalski, R. S. (1973), AQVAL/1 - Computer implementation of a variable valued logic system VL1 and examples of its application to pattern recognition, in: *Proceedings of the 1rst IJCPR*, pp. 3-17, Washington, D. C. (Michalski).

c 264. Michalski, R. S. (1973), Discovering classification rules using variable-valued logic system VL1, in: *Proc. Third IJCAI*, pp. 162-172.

c 265. Michalski, R. S. (1975), Variable-Valued Logic and its Applications to Pattern Recognition and Machine Learning, in: Rine, D.(Ed.) *Multiple-Valued Logic and Computer Science*, North-Holland.

c 266. Michalski, R. S. (1975), Synthesis of optimal and quasi-optimal variable-valued logic formulas, in: *Fifth Int. Symp. on Multiple-Valued Logic.*

c 267. Michalski, R. S. (1976), Learning by inductive inference, in: Simon, J. C.(Ed.) *Computer Oriented Learning Proceses*, pp. 321-337, Noordhoff Int. Pub. Co., Leyden, Netherlands (Proc. of the NATO Advanced Study Institute on Computer Oriented Learning Processes., Bonas, France. 1974).

c 268. Michalski, R. S. (1977), A system of programs for computer-aided induction: a summary, in: *IJCAI5*, pp. 319-320, Cambridge, Mass.

c 269. Michalski, R. S. (1978), Pattern recognition as knowledge-guided induction, Report #927, Department of Computer Science, University of Illinois at Urbana.

c 270. Michalski, R. S. (September 1980), Knowledge acquisition through conceptual clustering: a theoretical framework and an algorithm for partitioning data into conjunctive concepts, in: *Policy Analysis and Information Systems 4(3).*

c 271. Michalski, R. S. and Chilausky, R. L. (June 1980), Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis, in: *Policy Analysis and Information Systems 4(2)* (Special issue on knowledge acquisition and induction).

c 272. Michalski, R. S. (July 1980), Pattern recognition as rule-guided inductive inference, in: *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2(4)*, pp. 349-361.

c 273. Michalski, R. S. (July 1980), Inductive learning as rule-guided generalization and conceptual simplification of symbolic descriptions, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 274. Mihram, G. A. (1972), The modelling process, in: *IEEE Trans. on Systems, Man, and Cybernetics SMC-2(5)*, pp. 621-629.

o 275. Minicozzi, E. (1976), Some natural properties of strong-identification in inductive inference, in: *Theoretical Computer Science 2*, pp. 345-360.

or 276. Minsky, M. and Papert, S. (1969), *Perceptrons*, MIT Press, Cambridge, Mass.

c 277. Mitchell, T. M. (1977), Version Spaces: A candidate elimination approach to rule learning, in: *IJCAI5*, pp. 305-310, Cambridge, Mass.

c 278. Mitchell, T. M. (December 1978), Version Spaces: An approach to concept learning, Ph.D. thesis, Stanford University (also Stanford CS report STAN-CS-78-711, HPP-79-2).

oc 279. Mitchell, T. M. (August 1979), An analysis of generalization as a search problem, in: *Proceedings of IJCAI6*, Tokyo, Japan.

ch 280. Mitchell, T. M., Utgoff, P. E. and Banerji, R. B. (July 1980), Learning problem-solving heuristics by experimentation, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

co 281. Morgan, C. G. (1971), Hypothesis generation by machine, in: *Artificial Intelligence 2*, pp. 179-187, North-Holland.

co 282. Morgan, C. G. (September 1975), Automated hypothesis generation using extended inductive resolution, in: *Advance Papers of IJCAI4*, pp. 351-356, Tbilisi, USSR.

283. Mostow, D. J. and Hayes-Roth, F. (February 1979), Machine-aided heuristic programming: a paradigm for knowledge engineering, Rand N-1007-NSF, The Rand Corporation, Santa Monica, CA.

h 284. Mostow, D. J. and Hayes-Roth, F. (1979), Operationalizing heuristics: some AI methods for assisting AI programming, in: *IJCAI6*, pp. 601-609, Tokyo.

h 285. Mostow, D. J. (approx 1980), Mechanical operationalization of concepts and heuristics, Ph.D. thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA. (working paper).

h 286. Mostow, J. (July 1980), Machine-aided operationalization of advice, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 287. Nagel, E. (1963), Carnap's theory of induction, in: Schilpp, P. A.(Ed.) *The Philosophy of Rudolf Carnap*, pp. 785-825, Open Court Publishing Co., La Salle, Illinois.

h 288. Neches, R. (April 1979), Promoting self-discovery of improved strategies, in: *Annual conference of the American Educational Research Association*, San Francisco, CA. (also report CIP : 398, Psychology Department, Carnegie-Mellon).

h 289. Neches, R. (approx 1980), Heuristic Procedure Modification, unpublished Ph.D. thesis, Psychology Department, Carnegie-Mellon University, Pittsburgh, PA. (in preparation).

h 290. Neves, D. and Anderson, J. R. (approx 1980), Becoming Expert at a Cognitive Skill, Carnegie-Mellon University (in preparation).

h 291. Neves, D. M. (1978), A computer program that learns algebraic procedures, in: *Proceedings of the 2nd Conference on Computational Studies of Intelligence*, Toronto.

hp 292. Neves, D. M. (July 1980), Learning procedures from examples, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 293. Newell, A., Shaw, J. C. and Simon, H. A. (1969), A variety of intelligent learning in a general problem solver, in: Yovits and Cameron(Eds.) *Self Organizing Systems*, Pergamon Press, New York.

h 294. Newman, C. and Uhr, L. (1965), BOGART: A discovery and induction program for games, in: *20th Nat. Conf. ACM*, pp. 176.

r 295. Nilsson, N. J. (1965), *Learning Machines*, McGraw-Hill, New York.

g 296. Pao, T. and Carr, J. W. (1978), A solution of the syntactical induction-inference problem for regular languages, in: *Computer Languages 3*, pp. 53-64.

g 297. Pao, T. W. L. (1969), A Solution of the Syntactical Induction-Inference Problem for a Non-Trivial Subset of Context-Free Language, Report 70-19, The Moore School of Electrical Engineering., U. Pennsylvania.

o 298. Pearl, J. (1978), On the connection between the complexity and credibility of inferred models, in: *International Journal General Systems 4*, pp. 255-264, Gordon and Breach Science Publishers Ltd.

g 299. Pepe, J. (1967), A Procedure to Determine by Observation the State Diagram of a Turing Machine, M.S. thesis, Math. Dept., M.I.T., Cambridge, Mass.

p 300. Perry, M. (undated), Program optimization as a limiting process, Dept. Computer Science., U. Colorado, Boulder, Co.

c 301. Pikas, A. (1966), *Abstraction and Concept Formation*, Harvard University Press, Cambridge, Mass.

c 302. Pivar, M. and Gord, E. (1964), The LISP program for inductive inference on sequences, in: Berkeley and Bobrow(Eds.) *The Programming Language LISP: Its Operation and Appplications*, pp. 260-289, MIT Press, Cambridge, Mass.

c 303. Pivar, M. and Finkelstein, M. (1964), Automation, using LISP, of inductive inference on sequences, in: Berkeley and Bobrow(Eds.) *The Programming Language LISP: Its Operation and Applications*, pp. 125-136, MIT Press, Cambridge, Mass.

c 304. Plotkin, G. D. (1970), A note on inductive generalization, in: Meltzer, B. and Michie, D.(Eds.) *Machine Intelligence 5*, pp. 153-163, Edinburgh University Press, Edinburgh.

c 305. Plotkin, G. D. (1971), A further note on inductive generalization, in: Meltzer, B. and Michie, D.(Eds.) *Machine Intelligence 6*, pp. 101-124, Edinburgh University Press, Edinburgh.

p 306. Podnieks, K. M. (1974), Comparing various concepts of function prediction, Part 1, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 68-81, Latvian State U., Riga, U.S.S.R.

p 307. Podnieks, K. M. (1975), Comparing various concepts of function prediction, Part 2, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 35-44, Latvian State U., Riga, U.S.S.R.

p 308. Podnieks, K. M. (1975), Probabilistic synthesis of enumerated classes of functions, in: *Soviet Math Dokl 16*, pp. 1042-1045.

309. Podnieks, K. M. (1977), Computational complexity of prediction strategies, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, Latvian State U., Riga, U.S.S.R.

p 310. Podnieks, K. M. (1977), Probabilistic program synthesis, in: Barzdin(Ed.) *Theory of Algorithms and Programs*, pp. 57-88, Latvian State U., Riga, U.S.S.R.

c 311. Pollich, M. (approx 1980), Generalizing from noisy data using version spaces: empirical results, technical report, Department of Computer Science, Rutgers University, New Brunswick, N.J. (to appear).

o 312. Polya, G. (1954), *Mathematics and Plausible Reasoning*, Princeton University Press, Princeton, N.J.

o 313. Popper, K. (1968), *The Logic of Scientific Discovery*, Harper and Row, New York (2nd edition).

c 314. Popplestone, R. J. (1970), An experiment in automatic induction, in: Meltzer, B. and Michie, D.(Eds.) *Machine Intelligence 5*, pp. 204-215, Edinburgh University Press, Edinburgh.

o 315. Pudlak, P. (1975), Polynomially complete problems in the logic of automated discovery, in: *Lecture Notes in Computer Science 32*, pp. 358-361, Springer-Verlag.

o 316. Pudlak, P. and Springsteel, F. N. (1979), Complexity in mechanized hypothesis formation, in: *Theoretical Computer Science 8*, pp. 203-225.

o 317. Putnam, H. (1963), Degree of confirmation and inductive logic, in: Schilpp, P. A.(Ed.) *The Philosophy of Rudolph Carnap*, pp. 761-783, Open Court Publishing Co., La Salle, Illinois.

o 318. Putnam, H. (1975), Probability and confirmation, in: *Mathematics, Matter and Method*, Cambridge U. Press (Originally appeared in 1963 as a Voice of America lecture).

319. Quinlan, J. R. (1979), Induction over large data bases, Report HPP-79-14, Heuristic Programming Project, Stanford University.

320. Quinlan, J. R. (1979), Discovering rules from large collections of examples: a case study, in: Michie, D.(Ed.) *Expert Systems in the Micro Electronic Age*, Edinburgh University Press, Edinburgh.

c 321. Quinlan, J. R. (April 1980), Semi-autonomous acquisition of pattern-based knowledge, in: *Australian Computer Bulletin* (also to appear in Infotech State of the Art Report on Expert Systems).

p 322. Quinlan, J. R. (July 1980), Inductive inference as a tool for the construction of high-performance programs, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 323. Rescher, N. (1970), *Scientific Explanation*, The Free Press, New York.

324. Riesbeck, C. (July 1980), Failure-driven reminding as a basis for learning, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

325. Rosenblatt, F. (1959), Perceptual generalization over transformation groups, in: *Self Organizing Systems*, Permagon Press, London.

326. Rychener, M. D. (August 1980), Approaches to knowledge acquisition: the instructable production system project, in: *AAAI1*, pp. 228-230, Stanford, CA. (expanded version in preparation).

o 327. Salmon, W. C. (1966), *The Foundations of Scientific Inference*, University of Pittsburgh Press, Pittsburgh, PA.

hr 328. Samuel, A. L. (1963), Some studies in machine learning using the game of checkers, in: Feigenbaum, E. A. and Feldman, J.(Eds.) *Computers and Thought*, pp. 71-105, McGraw-Hill, New York.

hr 329. Samuel, A. L. (1967), Some studies in machine learning using the game of checkers II - recent progress, in: *IBM Journal of Research and Development 11(6)*, pp. 601-617.

p 330. Sato, M. (1979), Towards a mathematical theory of program synthesis, in: *Proc. sixth Int. Joint Conference on Artificial Intelligence*, pp. 757-762.

331. Savage, L. (1972), *The Foundations of Statistics*, Dover Publications, New York (second edition).

o 332. Savage, L. J. (1962), *The Foundations of Statistical Inference*, Wiley, New York.

o 333. Schlipp, P. (1963), *Library of Living Philosophers: The Philosophy of Rudolph Carnap*, Open Court Publishing Co., LaSalle, IL.

p 334. Schubert, L. K. (1974), Iterated limiting recursion and the program minimalization problem, in: *JACM 21(3)*, pp. 436-445.

p 335. Schubert, L. K. (1974), Representative samples of programmable functions, in: *Information and Control 25*, pp. 30-44.

g 336. Shamir, E. (1962), A remark on discovery algorithms for grammars, in: *Information and Control 5*, pp. 246-251.

337. Shannon, C. E. (1951), Prediction and entropy of printed English, in: *Bell Systems Technical Journal 30*, pp. 50-64.

p 338. Shaw, D., Swartout, W. and Green, C. (1975), Inferring LISP programs from example problems, in: *Fourth International Joint Conference on Artificial Intelligence*, pp. 351-356.

p 339. Shaw, D. E., Swartout, W. R. and Green, C. C. (September 1975), Inferring LISP programs from examples, in: *Proceedings of IJCAI4*, pp. 351-356, Tbilisi, USSR.

r 340. Shimura, M. (1978), Learning procedures in pattern classifiers- introduction and survey, in: *Proc. Fourth Int. Joint Conference on Pattern Recognition*, pp. 125-138.

g 341. Siklossy, L. (1972), Natural Language Learning by Computer, in: *Representations and Meanings: Experiments with Information Processing*.

p 342. Siklossy, L. and Sykes, D. (1975), Automatic program synthesis from example problems, in: *Fourth International Joint Conference on Artificial Intelligence*, pp. 268-273.

c 343. Simon, H. A. and Kotovsky, K. (1963), Human Acquisition of Concepts for Sequential Patterns, in: *Psychological Review 70*, pp. 534-546.

o 344. Simon, H. A. and Lea, G. (1974), Problem solving and rule induction: a unified view, in: Gregg, L. W.(Ed.) *Knowledge and Cognition*, pp. 105-127, Lawrence Erlbaum Associates, Potomac, Maryland.

o 345. Simon, H. A. (1977), Models of Scientific Discovery, in: Hintikka, J.(Ed.) *Synthese Library*, Reidel Pub. Co.

346. Smith, C. H. (1979), Hierarchies of Identification Criteria for Mechanized Inductive Inference, Ph.D. dissertation, State U. of New York at Buffalo, New York.

g 347. Smith, D. E. (July 1980), FOCUSER, a strategic interaction paradigm for language acquisition, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

p 348. Smith, D. R. (September 1980), A Survey of the Synthesis of LISP Programs from Examples, technical report, Bonas, France.

o 349. Smith, R. G., Mitchell, T. M., Chestek, R. A. and Buchanan, B. G. (1977), A model for learning systems, in: *IJCAI5*, pp. 338-343, Cambridge, Mass.

h 350. Smith, R. L. Jr. (August 1980), Modelling student acquisition of problem solving skills, in: *AAAI80*, pp. 221-223.

h 351. Smith, R. L. Jr. (July 1980), Modelling student acquisition of problem-solving skills: the student's interpretation of the teaching environment, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

352. Solmonoff, R. J. (1957), An inductive inference machine, in: *IRE Convention Record*, pp. 56-62.

g 353. Solmonoff, R. J. (1959), A new method for discovering the grammars of phrase structured languages, in: *Trans. Int. Conf. on Information Processing*, UNESCO House, Paris.

354. Solmonoff, R. J. (1962), Training sequences for mechanized induction, in: Yovits, M., Jacobi, G. and Goldstein, G.(Eds.) *Self-organizing Systems*, pp. 425-434, Spartan Books, Washington, D.C.

o 355. Solmonoff, R. J. (1964), A formal theory of inductive inference, in: *Information and Control 7*, pp. 1-22, 224-254.

o 356. Solmonoff, R. J. (1975), Inductive Inference theory- a unified approach to problems in pattern recognition and artificial intelligence, in: *Fourth International Joint Conference on Artificial Intelligence*, pp. 274-280.

o 357. Solmonoff, R. J. (1978), Complexity-based induction systems: comparisons and convergence theorems, in: *IEEE Trans on Information Theory IT-24(4)*, pp. 422-432.

o 358. Solomonoff, R. (1975), Inductive inference theory - a unified approach to problems in pattern recognition and artificial intelligence, in: *IJCAI4*, pp. 274-280, Cambridge, Mass.

h 359. Soloway, E. M. and Riseman, E. M. (1976), Mechanizing the common-sense inference of rules which direct behavior, in: *Proc. AISB Summer Conf*, U. Edinburgh.

c 360. Soloway, E. M. and Riseman, E. M. (1977), Levels of pattern description in learning, in: *IJCAI5*, pp. 801-811, Cambridge, Mass.

361. Soloway, E. M. and Riseman, E. M. (1978), Knowledge-directed learning, in: Waterman, D. A. and Hayes-Roth, F.(Eds.) *Pattern-Directed Inference Systems*, Academic Press, New York.

362. Soloway, E. M. (1978), Learning = Interpretation + Generalization: A Case Study in Knowledge-Directed Learning, Ph.D. thesis, report COINS TR-78-13, Computer and Information Science Dept., U. Mass. at Amherst.

c 363. Srinivasan, C. V. and Sandford, D. M. (1980), Knowledge Based Learning Systems: Part I: A Proposal for Research, Part II: An Introduction to the Meta-Theory, Part III: Logical Foundations, DCS-TR-89, Department of Computer Science, Rutgers University, New Brunswick, N.J.

c 364. Srinivasan, C. V. (July 1980), Knowledge based learning systems, an example, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA.

o 365. Stegmuller, W. (1976), *The Structure and Dynamics of Theories*, Springer-Verlag.

c 366. Stepp, R. (November 1978), The investigation of the UNICLASS inductive program AQ7UNI and User's Guide, Report 949, Department of Computer Science, University of Illinois, Urbana, Illinois.

c 367. Stepp, R. (1979), Learning without negative examples via variable valued logic characterizations: the uniclass inductive program AQ7UNI, UIUCDCS-R-79-982, Dept. Computer Science., U. Illinois, Urbana.

c 368. Stepp, R. (July 1980), Learning from observation: experiments in conceptual clustering, in: *Proceedings of the Workshop on Machine Learning*, Carnegie-Mellon University, Pittsburgh, PA. (not in collection though listed).

h 369. Stolfo, S. J. and Harrison, M. C. (January 1979), Automatic discovery of heuristics for non-deterministic programs, Technical report 007, Courant Institute.

o 370. Strong, J. (1976), The infinite ballot box of nature: De Morgan, Boole, and Jevons on probability and the logic of induction, in: *Philosophy of Science Association-1976*, pp. 197-211.

p 371. Summers, P. D. (1977), A methodology for LISP program construction from examples, in: *Journal of the ACM 24*, pp. 161.

h 372. Sussman, G. J. (August 1973), A Computational Model of Skill Acquisition, MIT AI-TR-297, Cambridge, Mass.

h 373. Sussman, G. J. (1975), *A Computer Model of Skill Acquisition*, American Elsevier, New York.

o 374. Tornebohm, H. (1966), Two measures of evidential strength, in: Hintikka, J. and Suppes, P.(Eds.) *Aspects of Inductive Logic*, pp. 81-95, North-Holland, Amsterdam.

g 375. Trakhtenbrot, B. and Barzdin, J. A. (1973), *Finite Automata: Behavior and synthesis*, North-Holland.

376. Trakhtenbrot, B. (1975), On problems solvable by successive trials, in: *Lecture Notes in Computer Science 32*, pp. 125-137, Springer-Verlag.

377. Tsypkin, Y. Z. (1973), *Foundations of the Theory of Learning Systems*, Academic Press, New York (Translated by Z. L. Nikolic).

r 378. Uhr, L. and Vossler, C. (1963), A pattern-recognition program that generates, evaluates, and adjusts its own operators, in: Feigenbaum, E. A. and Feldman, J.(Eds.) *Computers and Thought*, pp. 251-268, Mc-Graw Hill, New York.

ro 379. Uhr, L. (1973), *Pattern Recognition, Learning and Thought*, Prentice-Hall, Englewood Cliffs, New Jersey.

g 380. Van der Mude, A. and Walker, A. (1978), On the inference of stochastic regular grammars, in: *Information and Control 38*, pp. 310-329.

381. VanLehn, K. (1980), On the representation of procedures in repair theory, technical report, Learning Research and Developement Laboratory, University of Pittsburgh, Pittsburgh, PA.

c 382. VanLehn, K. (approx 1980), Algorithms for learning by examples, technical report, Xerox Palo Alto Research Center, Palo Alto, CA. (forthcoming).

383. VanLehn, K. and Friend, J. (1980), Results from DEBUGGY: An analysis of systematic subtraction errors, technical report, Xerox Palo Alto Research Center, Palo Alto, CA.

g 384. Veelenturf, L. P. J. (1978), Inference of sequential machines from sample computations, in: *IEEE Trans. on Computers C-27*, pp. 167-170.

c 385. Vere, S. A. (1975), Induction of concepts in the predicate calculus, in: *IJCAI4*, pp. 281-287, Tbilisi, USSR.

h 386. Vere, S. A. (1977), Induction of relational productions in the presence of background information, in: *IJCAI5*, pp. 349-355, Cambridge, Mass.

h 387. Vere, S. A. (1978), Inductive learning of relational productions, in: Waterman, D. A. and Hayes-Roth, F.(Eds.) *Pattern-Directed Inference Systems*, Academic Press, New York.

h 388. Vere, S. A. (September 1980), Multilevel counterfactuals for generalizations of relational concepts and productions, in: *Artificial Intelligence 14(2)*, pp. 138-164.

o 389. Watanabe, S. (1960), Information-theoretic aspects of inductive and deductive inference, in: *IBM Journal of Research and Development 4(2)*, pp. 208-231.

h 390. Waterman, D. (1968), Machine Learning of heuristics, Ph.D. dissertation and report No. CS118, AI 74, Stanford U.

h 391. Waterman, D., Anderson, R. H., Hayes-Roth, F., Klahr, P., Martins, G. and Rosenschein, S. J. (May 1979), Design of a rule-oriented system for implementing expertise, N-1158-1-ARPA, The Rand Corporation.

h 392. Waterman, D. A. (1970), Generalization learning techniques for automating the learning of heuristics, in: *Artificial Intelligence 1(1/2)*, pp. 121-170.

h 393. Waterman, D. A. (1975), Adaptive production systems, in: *IJCAI4*, pp. 296-303, MIT, Cambridge, Mass.

g 394. Wharton, R. M. (1974), Approximate language identification, in: *Information and Control 26*, pp. 236-255.

g 395. Wharton, R. M. (1977), Grammar enumeration and inference, in: *Information and Control 33*, pp. 253-272.

p 396. Wiehagen, R. and Liepe, W. (1974), Characteristic properties of recognizable classes of recursive functions, in: *Elektronische Informationsverarbeitung und Kybernetik 12*, pp. 421-438 (in German).

p 397. Wiehagen, R. (1974), Inductive Inference of recursive functions, in: *Lecture Notes in Computer Science 32*, pp. 462-464, Springer-Verlag.

p 398. Wiehagen, R. (1976), Limes-erkennung rekursiver funktionen durch spezielle straegen, in: *Elektronische Informationsverarbeitung und Kybernetik 12*, pp. 93-99 (in German).

g 399. Wiehagen, R. (1977), Identification of formal languages, in: *Lecture Notes in Computer Science 53*, pp. 571-579, Springer-Verlag.

p 400. Wiehagen, R. and Jung, H. (1977), Rekursionstheoretische charakterisierung von erkennbaren klassen rekursiver funktionen, in: *Elektronische Informationsverarbeitung und Kybernetik 13*, pp. 385-397.

o 401. Wiehagen, R. (1978), Characterization problems in the theory of inductive inference, in: *Lecture Notes in Computer Science 62*, pp. 494-508, Springer-Verlag.

o 402. Windeknecht, T. G. (1964), A Theory of Simple Concepts with Applications, Ph.D. dissertation, Case Institute of Technology.

c 403. Winston, P. H. (September 1970), Learning structural descriptions from examples, MIT AI-TR-231, MIT, Cambridge, Mass.

404. Winston, P. H. (1979), Learning by Understanding Analogies, Memo 520, M.I.T. AI Lab, Cambridge, Mass.

Category g - Grammars, Languages, Automata

Includes references whose inductions are in the form of a grammar. Induction of automata and language aquisition references are included here due to the close relation between grammars, languages and automata, even though the approaches used in the three areas may not be directly related.

{7, 12, 32, 33, 41, 44, 45, 48, 49, 98, 99, 101, 102, 103, 104, 106, 113, 120, 122, 124, 125, 126, 127, 146, 147, 149, 150, 153, 156, 160, 162, 165, 174, 192, 200, 201, 204, 220, 225, 259, 296, 297, 299, 336, 341, 347, 353, 375, 380, 384, 394, 395, 399}

Category r - Algebraic Discriminants (Pattern Recognition)

Includes references for induction of discriminants expressed as algebraic, or more generally, analytic relations between real-valued features of instances.

{57, 86, 107, 115, 147, 148, 151, 211, 212, 213, 234, 235, 241, 243, 244, 276, 295, 328, 329, 340, 378, 379}

Category c - Structural Discriminants (Concept Formation)

Includes references for induction of structural descriptions to characterize a class of instances. Such descriptions typically represent induced concepts in terms of components of the instances, component properties and relations among the components. Some approaches use the constructs of a logical language such as a predicate calculus, or a variable-valued logic, while others use tailored languages suitable to the application.

{8, 13, 20, 21, 22, 24, 25, 26, 27, 28, 30, 58, 67, 69, 70, 78, 92, 93, 94, 95, 96, 108, 109, 110, 111, 112, 115, 117, 128, 132, 157, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 193, 197, 202, 203, 204, 205, 206, 228, 246, 247, 248, 249, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 277, 278, 279, 280, 281, 282, 301, 302, 303, 304, 305, 311, 314, 321, 343, 360, 363, 364, 366, 367, 368, 382, 385, 403}

Category p - Functions and Procedures (Automatic Programming)

Includes references for induction of functions and procedures from input-output pairs or traces.

{2, 4, 14, 23, 31, 33, 34, 35, 37, 38, 39, 40, 46, 47, 50, 51, 52, 55, 74, 119, 123, 138, 158, 159, 168, 173, 208, 209, 210, 217, 219, 221, 222, 223, 224, 233, 253, 292, 300, 306, 307, 308, 310, 322, 330, 334, 335, 338, 339, 342, 348, 371, 396, 397, 398, 400}

Category h - Rules, Heuristics, Strategies

Includes references for induction of situation-action rules. Induction of grammars can be viewed as a specialization of this category, which we distinguish separately in category "g". We include learning of heuristics and strategies here since these are often couched in such rules.

{6, 9, 15, 16, 17, 22, 40, 56, 61, 62, 66, 67, 68, 69, 70, 71, 116, 118, 128, 130, 163, 164, 167, 182, 186, 187, 189, 190, 234, 235, 245, 257, 260, 280, 284, 285, 286, 288, 289, 290, 291, 292, 294, 328, 329, 350, 351, 359, 369, 372, 373, 386, 387, 388, 390, 391, 392, 393}

Category o - Overviews, Theories

Includes references on overviews, general theories and philosophies of learning and induction.

{1, 5, 6, 8, 9, 10, 28, 29, 36, 42, 53, 54, 55, 64, 65, 72, 79, 80, 81, 82, 83, 88, 89, 100, 129, 131, 142, 157, 166, 167, 170, 171, 172, 191, 194, 195, 196, 198, 199, 214, 218, 231, 236, 237, 238, 239, 240, 242, 252, 253, 254, 255, 256, 258, 262, 274, 275, 276, 279, 281, 282, 287, 293, 298, 312, 313, 315, 316, 317, 318, 323, 327, 332, 333, 344, 345, 349, 355, 356, 357, 358, 362, 365, 370, 374, 379, 389, 401, 402}