

AD-A139 685

PROCEEDINGS OF THE ARMY CONFERENCE ON APPLICATION OF
ARTIFICIAL INTELLIGENCE (U) BATTELLE WASHINGTON
OPERATIONS DC B J TULLINGTON 31 JAN 84

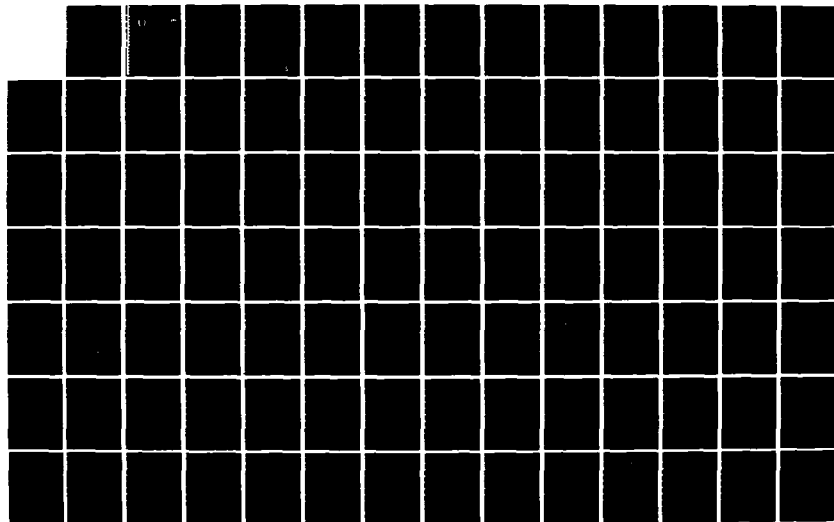
1/

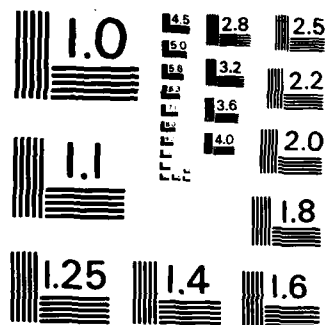
UNCLASSIFIED

DAG29-81-D-0100

F/G 5/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

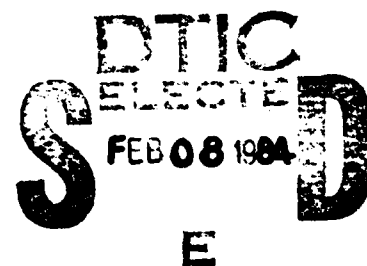
AD A139685



Proceedings of
**The Army
Conference on
Application of
Artificial Intelligence to
Battlefield Information
Management**



April 20, 21, and 22, 1983
at
U.S. Navy Surface Weapons Center
White Oak, Maryland



Sponsored by:

U.S. Army Electronics Research
and Development Command
Adelphi, Maryland

U.S. Army Research Office
Durham, North Carolina

DTIC FILE COPY

84 02 08 028

COMPONENT PART NOTICE

THIS PAPER IS A COMPONENT PART OF THE FOLLOWING COMPILATION REPORT:

(TITLE): Proceedings of the Army Conference on Application of Artificial Management
Held at White Oak, Maryland on 20, 21, and 22, April 1983.

(SOURCE): Battelle Columbus Laboratories, Battelle Washington Operations,
2030 M St., Washington, D. C. 20036

TO ORDER THE COMPLETE COMPILATION REPORT USE AD-A139 685.

THE COMPONENT PART IS PROVIDED HERE TO ALLOW USERS ACCESS TO INDIVIDUALLY AUTHORED SECTIONS OF PROCEEDINGS, ANNALS, SYMPOSIA, ETC. HOWEVER, THE COMPONENT SHOULD BE CONSIDERED WITHIN THE CONTEXT OF THE OVERALL COMPILATION REPORT AND NOT AS A STAND-ALONE TECHNICAL REPORT.

THE FOLLOWING COMPONENT PART NUMBERS COMPRISE THE COMPILATION REPORT:

AD#: P003 017	TITLE: Artificial Intelligence and Robotics for Military Systems.
P003 018	Opening Remarks on Artificial Intelligence.
P003 019	Navy AI (Artificial Intelligence) Programs--With Emphasis on Applications.
P003 020	Expert Systems for Understanding Remotely Sensed Images.
P003 021	The AI (Artificial Intelligence) Research Environment at the U.S. Army Engineer Topographic Laboratories.
P003 022	The Use of AI (Artificial Intelligence) in Target Classification.
P003 023	AI (Artificial Intelligence) Context Analysis for Automatic Target Recognition.
P003 024	Integrated DSS (Decision Support System) Development Tools for Micro Computers.
P003 025	Applications of Artificial Intelligence to Tactical Operations.
P003 026	Expert Systems for Intelligence Fusion.
P003 027	An Architecture for the Application of AI (Artificial Intelligence) Techniques To Threat Warning.
P003 028	An Overview of the Applicability and Use of Artificial Intelligence Techniques To The Processing of Communications and Noncommunications Signals.
P003 029	A Natural Language Interactive Computer System.
P003 030	Expert System for Tactical Indications and Warning (I&W) Analysis.
P003 031	Syntax Problems With Speech Recognition In Simulator Training Systems.
P003 032	Technological Assessment of Future Battlefield Robotic Applications.
P003.033	Autonomous Vehicle Control Using AI (Artificial Intelligence) Techniques.
P003 034	Expert Systems.
P003 035	Knowledge Acquisition and Evaluation Within Expert Systems.
P003 036	Experimental Logic and The Automatic Analysis of Algorithms.

COMPONENT PART NOTICE (CON'T)

AD#:

TITLE:

DTIC	
WMA	☒
LAB	☐
Unneeded	☐
Elimination	☐
Distribution/	
Availability Codes	
Avail and/or	Special
Dist	
A-1	

This document has been approved for public release and sale; its distribution is unlimited.

DTIC
ELECTRONIC
S APR 4 1984
A

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM																		
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER																		
4. TITLE (and Subtitle) Proceedings of the US Army Conference on Application of Artificial Intelligence to Battlefield Information Management, April 20, 21, and 22, 1983		5. TYPE OF REPORT & PERIOD COVERED Proceedings Oct 82 - Jan 84																		
7. AUTHOR(s) Bernard J. Tullington (Editor)		6. PERFORMING ORG. REPORT NUMBER																		
9. PERFORMING ORGANIZATION NAME AND ADDRESS Battelle Columbus Laboratories Battelle Washington Operations 2030 M St., N.W. Washington, D.C. 20036		8. CONTRACT OR GRANT NUMBER(s) DAAG29-81-D-0100 Delivery Order 0418																		
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Research Office P.O. Box 12211 Research Triangle Park, N.C. 27709		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Scientific Support Program																		
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Hq., US Army Electronics Research & Development Command, 2800 Powder Mill Road, Adelphi, MD 20783		12. REPORT DATE January 31, 1984																		
		13. NUMBER OF PAGES 301																		
		15. SECURITY CLASS. (of this report) Unclassified																		
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE																		
16. DISTRIBUTION STATEMENT (of this Report) Unlimited																				
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Unlimited																				
18. SUPPLEMENTARY NOTES <i>Artificial Intelligence (AI)</i>																				
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)																				
<table border="0"> <tr> <td>Artificial Intelligence</td> <td>Intelligence Fusion</td> <td>Knowledge Base Systems</td> </tr> <tr> <td>Battlefield Information Management</td> <td>Natural Language</td> <td>Robotics</td> </tr> <tr> <td>Tactical Operations</td> <td>Signal Processing</td> <td>Knowledge Acquisition</td> </tr> <tr> <td>Expert Systems</td> <td>Speech Recognition</td> <td>Knowledge Engineering</td> </tr> <tr> <td>Multisensor Target Identification</td> <td>Image Understanding</td> <td>Pattern Recognition</td> </tr> <tr> <td></td> <td></td> <td>Vehicle Control</td> </tr> </table>			Artificial Intelligence	Intelligence Fusion	Knowledge Base Systems	Battlefield Information Management	Natural Language	Robotics	Tactical Operations	Signal Processing	Knowledge Acquisition	Expert Systems	Speech Recognition	Knowledge Engineering	Multisensor Target Identification	Image Understanding	Pattern Recognition			Vehicle Control
Artificial Intelligence	Intelligence Fusion	Knowledge Base Systems																		
Battlefield Information Management	Natural Language	Robotics																		
Tactical Operations	Signal Processing	Knowledge Acquisition																		
Expert Systems	Speech Recognition	Knowledge Engineering																		
Multisensor Target Identification	Image Understanding	Pattern Recognition																		
		Vehicle Control																		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) These proceedings serve as a record of the conference which had the objective of bringing together practitioners, theoreticians, and potential users of AI to focus their efforts on the present accomplishments and existing technology base that is amenable to exploitation and further research in the application of AI to battlefield information management. Approximately 500 attendees from Government, Industry and Academia gathered to participate in the nine sessions. Twenty-five papers are presented in these proceedings in the areas of ongoing defense programs, intelligence fusion, signal processing, robotics, expert systems application, and natural (over)																				

Abstract continued:

languages. The program agenda and list of attendees is also included.

TABLE OF CONTENTS

	Page
FOREWORD	iii
INTRODUCTION	v
Artificial Intelligence and Robotics for Military Systems	1
Opening Remarks on Artificial Intelligence	7
Navy AI Programs--With Emphasis on Applications	11
Expert Systems for Understanding Remotely Sensed Images	17
The AI Research Environment at the U.S. Army Engineer Topographic Laboratories	37
The Use of AI in Target Classification	45
AI Context Analysis for Automatic Target Recognition	51
Integrated DSS Development Tools for Micro Computers	63
Applications of Artificial Intelligence to Tactical Operations	81
Expert Systems for Intelligence Fusion	101
An Architecture for the Application of AI Techniques To Threat Warning	117
An AI Approach to Multisensor Target Identification	125
Attempts At Applying AI to Situation Analysis	127
An Overview of the Applicability and Use of Artificial Intelligence Techniques To The Processing of Communications and Noncommunications Signals	129
A Natural Language Interactive Computer System	135
Expert System for Tactical Indications and Warning (I&W) Analysis	145
Syntax Problems With Speech Recognition In Simulator Training Systems	161
A Message Understanding Front End for a Knowledge-Based Threat Warning System	167

TABLE OF CONTENTS (continued)

	Page
Technological Assessment of Future Battlefield Robotic Applications	169
Autonomous Vehicle Control Using AI Techniques	181
Spatial Reasoning for Mobility and Manipulation	191
Expert Systems	193
Knowledge Acquisition and Evaluation Within Expert Systems	207
Applications of Knowledge Engineering	215
Experimental Logic and The Automatic Analysis of Algorithms	217
 APPENDIX A--PROGRAM AGENDA	 283
APPENDIX B--CONFERENCE ATTENDEES	291

FOREWORD

The rapid expansion of technology in such areas as communications, data processing, sensors and micro-miniaturization brought about the formulation of new tactical battlefield concepts based on dispersion and relative autonomy of small combat units. Such concepts lead to substantial increases in survivability and provide "force multipliers" in various tactical scenarios.

The enormous proliferation of the information flow stemming from the dispersion of combat units, as well as rapid growth of decision processes resulting from such a development require new approaches beyond the capabilities of the current state-of-the-art. One of the most promising approaches is based on the use of "expert systems" augmented by other tools of Artificial Intelligence (AI) such as "heuristic searches" of decision trees.

In recognition of those new realities, the US Army Electronics Research and Development Command (ERADCOM), which has paramount interest and mission in the area of battlefield information management, with support from the Army Research Office (ARO), organized a broadbased symposium, which articulated potential benefits of new AI concepts to such ERADCOM mission areas as signal processing, image understanding, fusion of sensor inputs, natural language understanding, automation and robotics, and possibly microelectronic system integration. The following proceedings, consisting of contributions from government, industry and academia, address various potential contributions of AI in the aforementioned mission areas.

The views, opinions, and/or findings contained in these proceedings are those of the authors and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

It is appropriate that certain acknowledgements be made at this point. The large attendance and close attentiveness of the audience during all nine sessions throughout the three day period was indicative of the hard work and leadership exhibited by the Session Chairmen and the expertise demonstrated by the individual authors. The Conference managers from HQ ERADCOM, the ARO, and Battelle Columbus Laboratories are most appreciative of their efforts. All the participants owe a debt of gratitude to Captain J. E. Fernandes (USN), the Commander of the Naval Surface Weapons Center (NSWC) at White Oak, who made their outstanding facilities available for this conference. Additionally, a special vote of thanks is in order to Ms. Vicki Mayhew and Ms. Leslie Karas of Battelle who kept track of us all and cheerfully handled the myriad details that accompany a conference of this scope and complexity. And lastly, we are grateful for the quiet efficiency of Mr. Brian Madden and his staff at NSWC who were the perfect host for our three day stay.



Accession For	
NTIS SERIAL	X
DTIC	
UNCLASSIFIED	
JAN 1977	
By	
Date	
Approved	
Dist	
A-1	

INTRODUCTION

Today, the Army faces some very difficult problems associated with training and retaining personnel to operate and maintain current high technology weapons systems. There is a growing awareness within the Army community that the field of Artificial Intelligence (AI) may offer solutions to these problems. Army researchers are also aware, however, that for many problems AI may offer only long-term solutions and, for some problems, may offer no solutions at all. Even when AI offers a solution some other field of scientific endeavor may offer a better one. Therefore, the Army is in the process of examining both the promise and pitfalls of AI. The Army Conference on Application of Artificial Intelligence to Battlefield Information Management, co-sponsored by the U.S. Army Electronics Research and Development Command and The U.S. Army Research Office, held April 20-22, 1983, was a part of that process.

The objective of this conference was to bring together practitioners, theoreticians, and potential users of AI to focus their efforts on the present accomplishments and the existing technology base that is amenable to exploitation, as well as to identify theoretical and practical issues requiring further research in the application of AI to battlefield information management. The purpose was to broadly educate the Army community on selected topics including image understanding, natural language understanding, expert systems, and robotics. The hope is that out of this will come a better understanding of where AI can be successfully employed in solving Army problems, where it should not be employed, and which basic research areas should be supported by the Army.

Artificial Intelligence techniques are being increasingly applied in many fields of interest to the Army, including image processing, adaptive controls, automatic test/diagnostic equipment, training and simulation, and medicine. Currently, the most popular and most promising of the subdisciplines of AI are expert systems, natural language processing, scene analysis, and the related area of robotics; i.e., intelligent automatic machines. Researchers in these areas have achieved some successes, and commercial products are available utilizing this technology. Army researchers are currently examining many of these developments for possible application to Army needs.

Expert systems, currently the most popular area of AI, has had some successes in medical diagnosis (MYCIN and CADUCEUS), miner exploration (PROSPECTOR), chemical analysis (DENDRAL), and configuration of computer systems to customer specifications (RI). MACSYMA, a system for symbolic computation, derives from the first expert system, Moses' SIN (program for Symbolic INtegration). Basic research issues in expert systems include knowledge representation, especially where more than one expert or a number of diverse knowledge bases are involved, acquiring and validating expertise (knowledge derived from experts rather than from textbooks), knowledge evaluation (is the knowledge to be added useful, or at least not harmful?), system testing and validation, control and search methods, and reasoning and inference methods.

Natural language processing is also currently a popular area of AI since it aims to produce the ultimate man-machine interface: voice input/output in a human, rather than a computer, language. Natural language processing also includes processing of natural language in written form (such as that typed on a computer terminal). Successes in this area have been harder to come by and are on a smaller scale than those in expert systems. Winograd's SHRDLU, a robot which operated in the Blocks World, conversed with a user in

natural language and was able to accept commands typed at the keyboard in English and explain the reasons for its moves. Speech recognition systems, such as HEARSAY and HARPY have had limited success and hardware speech recognizers are available but are also limited in vocabulary and must be trained to individual speakers. Signal processing is obviously an important part of speech recognition systems. Basic research issues in natural language processing include knowledge representation, reasoning, control and search methods, language and text analysis, domain modeling, task modeling, discourse modeling, grammars and parsing techniques, and special purpose machine architectures.

Image processing is a third very popular area of AI which proposes to give the machine eyes. In this area, non-AI researchers have generally taken a signal processing approach while AI researchers have attempted to apply domain specific knowledge to the scene. Again, there have been limited successes, but there is still a long way to go. Machine vision, even more than speech recognition, will benefit from faster algorithms and special purpose architectures. Other basic research issues include knowledge representation and modeling, image extraction, control and search methods, concurrent processing, determination and use of surface properties such as color, texture, coatings, etc., knowledge acquisition/learning, sensing techniques and interfacing with robot planning systems.

Research in robotics aims at producing autonomous machines rather than simply automatic machines. In addition to its connection with machine vision research and design of mechanisms, basic research issues in robotics include reasoning, especially spatial reasoning, knowledge acquisition/learning, and nonlinear adaptive controls.

Expert systems potentially have a broad range of applications for military purposes, some of which are reported on in this volume. In fact, more than half of the conference presentations involved expert systems. Some of the potential applications are to image understanding (to which an entire session was devoted--see, for example, papers by Kandal, Spiessbach and Gilmore, and Leighty), signal processing (see papers by Chubb and Bonasso), simulation and training, automatic testing/diagnostics, and semi-automatic control systems. Natural language processing has application to user friendly, hands-off man-machine interfaces (see the paper by Biermann). Image/signal analysis has application to remote intelligence gathering and processing (see the paper by Kiremidjian, Lenat and Clarkson), automatic target recognizers/classifiers, passive ranging systems, and photo interpretation. Robotics has applications in manufacturing, remote intelligence gathering and performance of hazardous duties (see the papers by Tseng and Bullock, and Brownstein and Reidy). In addition, a number of papers dealt with problems, approaches and tools in AI applications (see papers by Brown, Cutler, Gevarter, Kant, Loveland and Whinston).

The purpose of these Proceedings is to serve as a record of the Conference and to stimulate interest in the application of AI to Army problems.

AD P003017

ARTIFICIAL INTELLIGENCE AND ROBOTICS FOR MILITARY SYSTEMS

Dr. Edith W. Martin
Deputy Under Secretary of Defense for
Research and Advanced Technology

We are particularly pleased to address the subject of AI since it is currently receiving broad interest throughout the country--there are no less than half-a-dozen symposia and reviews scheduled for this spring alone. AI may finally be ready to emerge from its long gestation period and play a major role in our defense systems.

Within the last few years we have witnessed an almost explosive expansion of the field of AI within various agencies of the Department of Defense. This has been sparked by the rapid growth in computer technology, by the development and better understanding of AI concepts, and by the progress that has been made in sensors and control devices.

It is gratifying to note that early DoD investments have helped to establish the scientific foundations upon which the present U.S. capabilities and thrusts in AI and Robotics are based. For example, ONR and DARPA have been supporting research in AI for over 20 years through the support of "Centers of Excellence" at several prominent universities. These centers have published extensively, hosted symposia for government and industry, and spawned technological innovations such as numerically-controlled machine tools.

Artificial Intelligence is difficult to define. You may know it when you see it, but formal definition is elusive. We

talk about abstractions, computer representations, and generalized decision making, all at levels of sophistication that would be considered intelligent in humans--and we have difficulty in making this precise.

There is a psychology program that some of you may be aware of that asks the on-line user--a psychology patient--a series of questions. The patients consider the questions both relevant and intelligent. Apparently, merely dealing with the questions alone can produce positive results without any answers provided. Is this AI?

Many attempts have been made to build automata that emulate human behavior. One of the early approaches was based upon analogy with the human brain. Electrical networks were designed with properties considered analogous to those of neural networks. We built devices that, in a limited but real sense, could remember, adapt, and learn.

But progress was slow, and other directions were taken. The 1950's and 1960's saw considerable effort applied to the areas of automatic programming, the translation of natural languages, game playing--first checkers and then chess--and image processing. We learned how to get machines to prove theorems in axiomatic systems.

Some inroads were made and then, in most areas, the problems became unmanageable. There was good progress, however, in those problems that were nicely bounded--like checkers and chess.

In the 1970's, the field of AI took off in two different but not disjointed directions. The success with nicely bounded problems gave us confidence to move into the area now called "Expert Systems" or "Knowledge-Based Systems". The second major direction is "Robotics".

Each expert system is oriented toward a narrow subject area that is well-understood and can be formalized in the computer. Expert systems can make inferences and draw conclusions in essentially the same manner as the human expert it models. In the expert system, conclusions are derived from general rules and relationships rather than from a pre-programmed decision path.

Knowledge-based systems are generally larger than expert systems--they can be multi-expert systems or general-capability inference-systems which require "binding" to a specific subject area to become an expert system. Knowledge-based or expert systems are not structured in the same manner as traditional data processing systems and in fact use somewhat different development aids. One concept that has significantly accelerated work in AI is that of the list processing language. One such language, called LISP, has become the language of discourse of AI. LISP was developed by Professor John McCarthy when he was at MIT.

Some of the major accomplishments in expert systems have been widely publicized. Examples are the geological discoveries aided by SRI's

"Prospector" system, Stanford's MYCIN for medical diagnosis, and DENDRAL, an expert system in molecular physics.

Robotics, which in some respects is an AI application area, but also a research area in its own right, combines the disciplines of expert systems, computer vision and sensing, pattern recognition, machine control of physical manipulation, and automobility. Major targets for robotics work include custom manufacturing and unmanned military systems.

AI Technology has potential application in many military mission areas such as intelligence gathering; processing and analysis; operations planning; command and control; tactical warfare; targeting; navigation; logistics; and fault protection. It would be very difficult to identify every DoD activity or mission area that could benefit from AI.

The important generic scientific issues which AI work is directly addressing are knowledge representation, knowledge acquisition, language, signal and image understanding, man-machine interaction, processing of multiple sensory data, and decision aids.

Currently, as research continues we are in a period of exploration and technology transfer. There are prototype AI efforts now on-going in the DoD in areas such as crisis warning and alerting, situation assessment, expert systems, fusion of sensor inputs, natural language understanding, automated computer programming, mission planning and scheduling, machine training, and man-machine interface in navigation. During 1982, the Department of Defense spent about \$18M in these areas and the total for 1983 is about \$28M, all at the 6.1, 6.2 and 6.3A levels. Through

these efforts, we will gradually begin to get a feel for the role that AI will play in defense systems.

At the same time, work must continue in the supporting technologies, for example, the structure of knowledge-based systems, highly parallel architectures and related HOL's, symbolic image interpretation, miniaturized sensors. Continued advances in VHSIC will provide high performance and large memory capacities in small packages that can help to make AI feasible for use in our weapons systems.

AI involves large-scale software and such software has presented the DoD with serious problems over the last decade. Our Ada* language effort is expected to help, but more is needed. Large scale system software is becoming more complex and less manageable every year. We have recently embarked on a new joint Service software initiative, building on Ada, called the STARS program--Software Technology for Adaptable, Reliable Systems. Ada's emphasis is on the programming and program design processes and its related environment. STARS focus will be on automated software tools that are applicable across the total software life cycle including specification, software system design, integration, testing and validation. The creation of technology and the establishment of practices that improve the software situation will directly improve our potential for progress in AI, since AI will involve some of the largest and most complex software systems ever built.

There is also a reverse implication. AI will help to improve the software environment by facilitating automatic or semi-automatic translation across levels of software specification--the

modern form of the old quest for automatic programming.

A major challenge to be met by the DoD AI community is the effective integration of multiple disciplines, e.g., graphics, pattern recognition, voice recognition, touch, sensors, controllers and the expert system.

The most ambitious goal may be to provide the commander with sophisticated decision aids employing AI. This goal will require more progress in optimizing decision paths in complex hierarchical branching trees using heuristic search and truncation techniques.

We should anticipate the maturing of AI to be accompanied by both management and socio-economic problems. The new technology will not only alter our way of life but also our mental perceptions. The use of robots in manufacturing will change the face of our industry and redefine the nature of labor-management relations. Robots will be an economic power. Also, certain creative processes, such as design and engineering, will be automated to some extent through AI technology and we will have to deal with this. Moreover, many management and planning functions may also be changed by AI. We can already see the impact of AI on training and education. This trend will provide much greater flexibility in the educational process.

The impact of AI on military technology and tactics may be tremendous. We may see greater autonomy, sophistication and dispersion of weapons systems and personnel.

The importance of AI is also recognized by the other major industrial

nations. The Japanese, for example, plan to invest about \$500M in their fifth-generation computer system project between now and 1990. This effort will place a heavy emphasis on AI. Similarly, the British and French governments are now supporting substantial efforts in AI technology and research. The French are emphasizing the social implications of this new technology.

What role should the DoD play in AI technology? We believe that the Department should be supportive and nurturing rather than directive and regulatory. DoD can provide support in five areas: (1) basic research, (2) support of higher education, (3) sharing of information, (4) standardization, and (5) mission related technology. I'll briefly elaborate each of those points.

1. Funding basic research: AI, including robotics technology, is not an isolated technology but rather the integration of a wide range of technologies and disciplines. Thus, much of the basic research sponsored by the DoD in many of these areas contributes to the creation of the technology base for AI, even if it is not specifically identified as such. Technology that is AI specific, such as the theory of expert or knowledge-based systems will, of course, be supported directly.
2. Support of higher education: We see DoD supporting higher education in two areas: (1) support of centers of excellence in order to advance the state of the art, and (2) providing seed money or other incentives encouraging the academic community to create curricula which will ultimately result in

the work force needed to support our progress in this area. Many of the DoD agencies now have programs oriented toward AI in the areas I've just discussed. In addition, we are in the final stages of selecting a few universities to serve as centers of excellence in AI and robotics in support of all of DoD. The current plan involves funding of each center at about \$1 million per year for four years.

3. Sharing of information or technology transfer: Another potential role for the DoD is the establishment of a central means for sharing information. Within the DoD community we presently have about 20 information analysis centers which serve as focal points for the various technological areas of interest to the DoD, to DoD contractors and to the private sector as well. In addition, Information Analysis Centers often provide administrative assistance to DoD committees in their respective technical areas and promote the exchange of technical information by the distribution of summary information. Within DoD we have discussed the establishment of an Information Analysis Center for AI and robotics, but no decision has been made yet.
4. Standardization: DoD has traditionally played a leadership role in voluntary standardization programs for U.S. industry. We should continue to do this in AI and robotics. Industry-wide standards will be needed in areas such as safety, terminology, mechanical and electrical interfaces, software interfaces, and man-machine interfaces.

We believe that the DoD role must be one of catalyst or facilitator in the establishment of these standards. Some seed money may be arranged to initiate these activities.

5. Mission related technology:
With regard to DoD's responsibility to insure national defense, we will do whatever is necessary at the technology level to gain those strategic and tactical advantages that AI and robotics may offer with respect to the execution of military missions, to avoid technological surprise by potential adversaries, and to promote efficiency and economy in the production of defense materials.

In closing, we would like to note that both DoD and individual military services have initiated programs in AI in recognition of its potential payoff.

Investments have been made and more are being planned in the future. For example, within the last few years, the Navy has established a very well equipped center for applied research in artificial intelligence located at NRL. This center has been active in organizing symposia with participation from universities, industry and DoD. DARPA in cooperation with the Air Force maintains centers of excellence at several universities. The Army has invested substantially in the use of AI in robotics. Many Service laboratories are pursuing work in their own interest areas.

With respect to funding, we have increased the AI budget by 65% from FY 82 to FY 83. We expect future budgets to include more increases.

It's an exciting area that now holds greater promise for near-term results. Consequently, this conference is quite important and very timely.

TAB 84-10

Unannounced

3 Apr 84

AD-D095 456 - AD-D095 494 = 39*

AD-A953 084 - AD-A953 102 = 19

A953109

AD-B955 416 - AD-B955 472 = 57

B955473

AD-B995 148 = 1

AD-C952 532 - AD-C952 555 = 24

140

* NBS DOCUMENTS

TAB 84-10

3 Apr 84

AD-A139 093 - AD-A139 601 = 509

AD-B080 584 - AD-B080 917 = 334

AD-C033 782 - AD-C033 869 = 88

} 422

AD-D010 910 - AD-D010 968 = 59

AD-P002 934 - AD-P003 016 = 83

AD-P200 000 - AD-P200 092 = 93

1166

OPENING REMARKS ON ARTIFICIAL INTELLIGENCE

BG Alan B. Salisbury

Director of the Special Task Force and Program Manager
Joint Tactical Fusion Program

INTRODUCTION

On behalf of Major General Paige, it gives me great pleasure to welcome you to this first major DOD symposium on the use of artificial intelligence in the management of battlefield information. I hope that you will find various presentations in the area of artificial intelligence to be given by a wide spectrum of speakers both interesting and germane to your own pursuits. Certainly as Director of the Special Task Force and Program Manager for the Joint Tactical Fusion Program, I am looking to the field of artificial intelligence with special interest for possible new techniques and solutions. I am sure that many of you will be able to apply some of the insights gained during the next three days to a variety of your own problems.

The current heightened interest within the Army in the field of artificial intelligence is in part the result of a rapid growth of various sensors on the battlefield and the need for their coordination. The amount of information acquired and handled by those sensors has already grown explosively in the last few years, and can be further expected to reach even higher levels if new tactical concepts embedded in the doctrine for the Air Land Battle 2000 and the VISTA Program are to materialize in the future.

In addition to the rapid growth of information processing, there is also a need for the accessibility of that

information at the various combat echelons in a much shorter time frame than is the case currently. "Real-Time" requirements become progressively more demanding as we move from echelons above Corps, through Corps and Division to Brigade, or even Battalion level. A more detailed analysis of the situation indicates that existing or anticipated computational and communication resources on the battlefield will be inadequate to meet those objectives. The obvious limitations on communications capacities drive us to looking for more and more intelligence closer to the sensors to avoid moving voluminous raw data through the pipes.

This article is missing

WHAT ARE ARMY EXPECTATIONS
FOR ARTIFICIAL INTELLIGENCE 38

Simply stated, we are hoping the artificial intelligence will provide new methodologies for handling information acquisition and processing problems which require less computational and communications resources than currently is the case, leading to getting the right information to the commander in time to act upon it.

One reason for this hope for artificial intelligence is based on its potential for more effective techniques of data compression. Such techniques may include the transformation of raw numerical data into domain of symbolic and semantic entities. This concept could be applied on all levels of battlefield information processing,

from individual sensors to complex adaptive networks and data processing modes. We believe that new computing formats based on symbolic calculus, with tools like LISP, for example, rather than simple number crunching, offer promising avenues in that direction.

ARTIFICIAL INTELLIGENCE IN SYSTEM INTEGRATION AND INFORMATION FUSION

One of the most difficult conceptual and technical areas is the fusion of the battlefield information acquired by multiple individual sensors. As Program Manager for the Joint Tactical Fusion Program where sensor inputs from many sources on the battlefield have to be integrated and fused, I am keenly aware of the difficulties in this area. In fact, even basic definitions, such as "fusion", when dealt with on a practical level, are not simple matters.

Although in the past few years we have learned a great deal about techniques of battlefield integration involving inputs from various sensor platforms, we believe that artificial intelligence could significantly contribute toward manageability of some of the analytical techniques. In particular, the systematic description of the system by means of knowledge frames and its integration by means of the "expert" system approach lend themselves toward simplifying our task.

As an example, current U.S. Army methods for collection/mission man-

agement and dissemination of intelligence (CM and D) require the cooperation of many people and lengthy preparation of plans. Tools utilized today include wall maps, file card boxes, and volumes of reference material. As a result of the slow manual preparation of CM and D plans, and delays in receiving and updating status information on sensors, rapid response to changing conditions cannot be provided and the task of command and control is therefore more difficult. The application of artificial intelligence to the CM and D process (both planning and dynamic recovery) will clearly improve the response time while providing increased flexibility and performance.

Artificial intelligence technology will be incorporated in the All Source Analysis System/Enemy Situation Correlation Element (ASAS/ENSCE) systems in an evolutionary manner which will eventually provide a highly automated, integrated operational system. The incorporation of advanced algorithms will require definition and acceptance of these automatic processes by operational elements of the services. These capabilities, when incorporated, will allow users to rapidly plan intelligence collection missions and provide dynamic recovery capabilities for unexpected events. A knowledge based or expert computer system could be developed incorporating the expertise of collection management that would provide accurate real-time information on sensor availability, capability, and information on the current task load. This use of artificial intelligence technology in the ASAS/ENSCE CM and D process would provide an automated ability to rapidly select and issue sensor/mission tasking messages from input requests for intelligence data. This is of course only one example drawn from my program.

OTHER ARMY APPLICATIONS OF ARTIFICIAL INTELLIGENCE

Although battlefield surveillance and target acquisition present some rather sophisticated challenges to the use of artificial intelligence, it is noteworthy that some complex problems of battlefield logistics can be also simplified by the use of artificial intelligence. In fact, some of the greatest payoffs of artificial intelligence as of this moment are in such areas as training, maintenance, operation of depots and so on. Still another area getting a lot of attention in the Army (where artificial intelligence is integrated in with machines) is in the area of robotics. There are several ongoing programs in robotics and there is a special session in this symposium dealing with sensory inputs such as image and signal processing, including natural speech processing and the fusion of various inputs into a format relevant for command and control.

ERADCOM ACTIVITY IN ARTIFICIAL INTELLIGENCE

In the next few days you will hear a number of papers describing a wide scope of activities in government, industry and academia. Let me therefore limit my concluding remarks to the brief description of U.S. Army Electronics Research and Development Command (ERADCOM) activities in the artificial intelligence area. These activities have been initiated as a result of the recommendations by the Defense Science Board and the Army Science Board to establish a

major thrust in the area of artificial intelligence and they have the full endorsement and support from ERADCOM Commander, MG Paige. Of primary concern to ERADCOM is the potential for the emerging artificial intelligence technology in the area of battlefield information and intelligence management. There is a wide variety of battlefield surveillance sensors including electro-optical, radar, COMINT, ELINT, acoustic, and NBC, which will be available in both ground based and airborne platforms. The need to derive, in real-time, battlefield target situation assessments for all elements of the battle force indicates the need for artificial intelligence solutions to the problems of target correlation, fusion, battlefield assessments and resource management. The primary ERADCOM laboratories which are currently engaged in some aspects of artificial intelligence are: Night Vision and Electro-Optics Laboratory (NVL); Signal Warfare Laboratory (SWL); Harry Diamond Laboratory (HDL); and Electronic Warfare Laboratory (EWL).

The programs of these ERADCOM laboratories will be covered in detail in a later session. The total funded ERADCOM effort in artificial intelligence is at the \$4.8M with approximately \$2M planned in FY 84.

CONCLUSION

I am sure that the scope and quality of this symposium will be a major landmark in the field of artificial intelligence in Department of Defense. I wish you all success in your work.

AD P003019

NAVY AI PROGRAMS--WITH EMPHASIS ON APPLICATIONS

Jude E. Franklin

Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory

ABSTRACT

The paper is an overview of the Navy AI programs. It lists the programs at all the Navy facilities and highlights the AI application research underway at the Navy Center for Applied Research in Artificial Intelligence.

INTRODUCTION

The Navy is beset by numerous problems stemming from the use of a growing volume of complex information and from the increasing complexity of its weapon systems. Decisions must be made faster than ever before and operational readiness must be maintained despite limitations on manpower and training. Artificial Intelligence (AI) technology holds much promise for solving some of these problems and is beginning to bear fruit. In order to solve these problems, the Navy has made major investments in the area of Artificial Intelligence (AI). The Office of Naval Research (ONR) has paid particular attention and funded research in AI for the last 20 years. This program in basic research has more recently been supplemented by applied programs at the Navy Center for Applied Research in Artificial Intelligence (NCARAI), Naval Ocean Systems Center (NOSC), Naval Surface Weapon Center (NSWC), Naval Weapons Center (NWC), the Naval Underwater Systems Center (NUSC), Naval Air Development

Center (NADC), Naval Ships Research & Development Center (NSRDC), and Naval Personnel Research & Development Center (NPRDC). This paper will briefly outline these Navy programs with emphasis on the Navy Center for Applied Research in Artificial Intelligence (NCARAI) that has been established at the Naval Research Laboratory. The Center develops and transitions AI technology into the Navy's Operational Units by demonstrations in the context of real Navy problems. One special application area is in Combat Management Information Processing where information is to be organized, analyzed and presented to a Commander. The Commander can then use the pertinent information and the AI decision support systems to make more accurate and timely decisions.

OVERVIEW OF NAVY PROGRAMS

BASIC PROGRAM

The Basic Research Program in AI is managed by ONR. The major

components of the 33 Basic Research Projects are:

- Robotics
- Knowledge Acquisition
- Automated Reasoning
- Man-Machine Interface
- Psychology
- Expert Systems
- Natural Language
- Crisis Alerting

This program is performed mainly in academia and at specific centers of excellence that have been sponsored by the Navy over many years. Participants include MIT, Stanford, U. Md., CMU, NYU, U. Pa., Columbia, Brown, SRI, U. Illinois, U. Utah, Yale, RPI, BBN, U.C. Irvine and U. Mass.

APPLIED PROGRAM

NCARAI

The general descriptions of the projects are listed below:

Task 1 - Expert System for Electronic Maintenance--This task is dedicated to enabling a Navy technician to troubleshoot and maintain complex Navy equipment using AI to emulate or surpass the heuristic search patterns and techniques that are used by expert technicians. The completion of this project will result in reduced system downtime and increased fleet readiness. Future work will involve the automatic generation of test code for automatic test equipment by means of an Expert System.

Task 2 - Message System Automation--In this task we are transitioning 6.1 research into a 6.2 project at the NCARAI. The goal is to allow a machine to disseminate a message to

appropriate recipients by computer understanding of the message information. This information is used as input to an expert system. Our efforts include updating message precedence in a dynamic environment, altering the decision maker in a more timely manner, and updating the data base. This task should help reduce operator overload.

Task 3 - Expert System for Decision Aids--This task is for the Marine Corps. We have developed a rule-based expert system to select a set of weapons for a given set of targets to produce maximum expected destruction. General purpose tools are being developed, as well, for other expert systems including one for target class identification for a Radar System. These tools will help the operator to cope with complex decisions that require a large amount of input data, and to make these decisions in a timely manner.

Task 4 - Multisensor Information Integration--This task will develop an experimental organization of automated knowledge-based specialists to integrate information from multiple sensors such as radar, sonar, ESM, intelligence and overhead surveillance. This information organization will support the commander and his staff in developing a sound tactical picture to guide decision making. This project will reduce operator overload and help the decision maker make accurate decisions with a better understanding of the surrounding environment.

Task 5 - Operational Planning--This task objective is to develop an expert consultant system to aid in naval warfare mission planning. The expert system will assist the planner by breaking the problem into smaller logical units or frames and then will reason about the best plan to accom-

plish the whole. This will allow plans to be created in a more timely fashion and will perform a series of checks on the final plan to determine conformance with the commander's overall mission.

The titles of the programs at the other Navy labs are listed below:

Naval Ocean Systems Center

- Confidence Mechanisms for Expert Systems (1982 start)
- Automated Information Collection for Fusion
- Message Analyzer and Disambiguator
- Tactical Situation Assessment (1981, none in 1982)
- Knowledge for Expert Systems
- Vocabulary Extensibility (1983)
- Communications Intelligence Expert System
- Mission Planning
- Voice Control Teleoperators
- Free Swimming Submersible

Naval Underwater Systems Center

- Cybernetics in Underwater Combat Control - Man Machine
- Information Management for Submarine Combat (No effort in FY83)
- Computer Aided Classifier -Submarine Sonar
- Voice Communications for Computer (6.1)

Naval Personnel Research and Development Center

- Training and Simulation (Steamer)
- Maneuvering Board Training
- Qualitative Graphical Interfaces for Quantitative Process Models

Naval Surface Weapons Center

- Increased Tracking Accuracy for Fire Control
- Adaptive Doctrine Management
- Heuristic Systems for Target Detection
- Signature Recognition and Decision Making for Underwater Explosions
- Natural Language Communication with Computers
- Computer Vision
- Pattern Recognition and Scene Analysis Applied to Target Detection

Naval Air Development Center

- Information Assessment for Airborne Command and Control (6.2)
- Decision Aids and Analysis for Airborne Command and Control

Naval Weapons Center

- Automatic Ship Classification (Harpoon)

Naval Training Equipment Center

- Voice Technology as an Instructors' Assistant (6.2)
- Automated Knowledge Acquisition (6.2)
- Individual Adaptive Training Systems (6.3)
- Adaptive Part Task Training (6.3)

DISCUSSION OF INDIVIDUAL PROJECTS

NCARAI

Natural Language

Application systems accepting natural language input have burgeoned in the

past few years. Most of these are interactive applications such as data base retrieval, document preparation, and command and control. The Navy Center for Applied Research in Artificial Intelligence has focussed on a different class of applications, involving the analysis of short reports from scientific and technical domains. There are many areas in which such reports are generated in large volume and there is a pressing need for an ability to process such documents automatically, in order to maintain data bases and gather statistics. We have developed techniques for analyzing such reports and have applied them to military messages concerning equipment failure.

What does it mean to understand such reports? For an individual application it means acting on the reports (creating a data base entry, computing a statistic) at a level comparable to human performance. For a range of applications, it means analyzing the text and converting it to a form that facilitates the development of application programs achieving this level of performance. Specifically, this requires identifying the information structures underlying a class of texts and automatically mapping the texts into these structures, thus making explicit the relations between constituents of the text.

Message System Automation — This task examines a class of Navy Messages (CASREPS) that are used to report equipment failures. The AI system will break the message into its parts of speech and the function of each within the sentence (subject, predicate, object, etc.). The system then takes the output of this parser and puts it into an information table to update data bases and to provide information for an expert system to update the message precedence and

dissemination. This research will allow automatic processing of classes of Navy Messages. Future transition applications include Rainforms and the 3M (Maintenance and Material Management) system. The reason this research can be applied to domains other than CASREPS is because the research is based on a grammar approach that determines the meaning of the message and does not just use key words as has been done in the past.

EXPERT SYSTEMS

Maintenance and Troubleshooting

Expert System for Electronic Maintenance--This task combines the use of automatic test equipment and AI. The AI research was originally done in academia and is being transitioned to this project from Stanford, MIT and Rutgers. The AI system will guide a technician through a complex block diagram and suggest where the next test should be performed. This test will be based on heuristic rules derived from other technicians, past failure rates, the size of the ambiguity group and conditional probabilities of the result of the test. The final system will interpret test results and show the technician how to perform the next test, by inferring functions from schematics and block diagrams. The system will also have displays from video discs and video tape recorders to show the technician how the test can be performed and where in the system the test must be made. The general techniques that have been developed will be applied to a specific military piece of equipment used to support C3. A very likely candidate is communications for the LAMPS MARK 3 System. This system will also be a candidate for a joint service effort

(sponsored by the Joint Directors of Laboratories, JDL) to be conducted and managed by NCARAI.

Combat Management

Expert System for Decision Aids--

General techniques in heuristic search, tree pruning, reduced operator question and answering as well as an explanation facility are being developed to aid the development of expert systems for decision support. The approach stresses the development of general purpose tools that can aid in the development of expert systems. This project is also providing the basic expert system for a new effort (sponsored by NAVELEX and NRL) in classification of images taken from new complex radars.

The first phase of this combat management expert system is the analysis of the effectiveness of each weapon-target allocation. The effectiveness of a weapon against a target is, by definition, the expected proportion of the target that would be destroyed if the weapon were fired at it. Effectiveness is the final output of a complex calculation that uses 55 factors of the weapon, target, and battlefield situation. Some examples are:

- Range and Position
- Personnel Readiness
- Counterfire Ability
- Resupply
- Ammunition Status
- Number of Tubes
- Maintenance Status

The second phase uses effectiveness to evaluate overall allocation plans. For any such plan, given the effectiveness

from phase one, the expert system computes the expected total destruction, D, for that plan. Since there is a finite, though large, number of possible plans, there is a unique maximum value of D. Any plan having that maximum D is an optimal plan. The System can find the optimal plan without looking at all possible plans because it uses a pruning algorithm to eliminate plans with values of D that are less than the maximum.

The user may choose to have the optimal plan generated or a plan that has a D value that is less than the maximum, that is, suboptimal. The trade-off is D value versus time. For example, the optimum solution for the allocation of eight weapons to 17 targets, in our research setting, took about 12 minutes (11 min. 43 sec.). A suboptimal allocation, D value, 98% optimal, consumed about seven seconds (6.75 sec.).

In addition to its thoroughness in considering the 55 factors in phase 1 and its assurance of optimal solution in phase 2, the expert system also has the following advantages for the user:

- User control of input
- Uses simple commands
- Easy error recovery
- Warns user of errors
- Interactive assistance

In phase one, the calculation of effectiveness for each weapon-target pair, this expert system moves significantly beyond the capabilities of its predecessor, MIFASS (Marine Integrated Fire and Air Support System), by the range of battlefield factors it considers. The current MIFASS implementation does take into account some restrictions directly related to the weapon and target, notably fire time, fire zones, and the availability

of ammunition. However, it ignores other important aspects of the situation, such as the combat readiness of personnel, resupply possibilities for various resources and weather, that can be decisive factors in a military engagement.

OPERATIONAL PLANNING

A system is being developed by transitioning the Meta Description System (MDS) from Rutgers to NCARAI. This system will help the mission planner reason about the problem from its goals, reduce it to various sub-goals or units and then translate the user's input into LISP programming code. The initial system has been modified and a specific example from NWP-II is being used to test the concept and the approach. The system allows the user to reason about the world even in a noisy environment. The project can be used in generation of op-orders but future efforts can be used to augment other NCARAI projects to solve problems where high level reasoning and understanding are required.

This task will result in an expert system that will reduce the time required to generate plans and op-orders and provide a means of alerting the user to possible conflicts in his proposed plans. Transition opportunities include the automatic generation of op-orders for Marine Corps' Amphibious Assault Landings.

DISTRIBUTED PROBLEM SOLVING - MULTISENSOR INFORMATION INTEGRATION

This AI approach uses a new architecture consisting of a society of communicating experts or specialists to integrate, correlate and determine the

validity of detections made from acoustic, radar, ESM, intelligence, and other sensors. This architecture was derived from past experience of AI systems such as Hearsay II, SIAP, SU/X. Analysis of critical issues such as message reporting granularity, functions for the individual expert system specialists, and knowledge representation techniques are being examined. The architecture will be encoded and examined in a skeleton form with simulated inputs. The initial system will prove the architecture will be encoded and examined in a skeleton form with simulated inputs. The initial system will prove the architecture by developing specialists such as Radar and Sonar Interface specialists, platform specialists, and report distribution specialists. After the architecture has been proven, it will be expanded and the entire network of communicating specialists will be tested with simulated and real inputs.

SUMMARY

The Navy has taken the first step to use Artificial Intelligence to solve the difficult problems facing the fleet today. The Basic Research Program is now augmented by an applied program that has been started at several Navy laboratories and centers. The Navy is also joining the other services in a JDL effort to use AI to solve difficult problems common to all of DOD.

EXPERT SYSTEMS FOR UNDERSTANDING REMOTELY SENSED IMAGES

Laveen N. Kanal, Barbara A. Lambird, and David Lavine
University of Maryland

ABSTRACT

This paper discusses some basic issues in the design of expert or knowledge-based systems for understanding remotely sensed images, with particular reference to cartographic feature extraction problems. The many types of knowledge that must be modelled in remote sensing and cartography and the characteristics distinguishing cartographic image processing from other image processing tasks are outlined. The problems of knowledge representation, knowledge-based search, fusion of information from multisensor sources and knowledge-bases, and the resolution of conflicting information are briefly treated. A distributed architecture and a control structure based on a parallel non-directional search algorithm are outlined. Finally, open problems are mentioned.

1.0 INTRODUCTION

Image understanding is central to achieving acceptable levels of performance by semi-automated and automated systems in several application domains of current interest, e.g., robotics, computer aided design (CAD), diagnosis using medical imagery, remote sensing, cartographic feature extraction, scene analysis, route planning and autonomous navigation.

Working with the U.S. Army Engineer Topographic Laboratory (Stockman, et al., 1981, Lambird, 1981 a, b) and others, it became evident to us that existing approaches to automatic and semi-automatic cartographic feature extraction from remotely sensed images did not work too well. Many problems were encountered including the sheer amount of information in

each image which taxed the memory and processing capabilities of the computers. The ambiguous and contradictory information which is present in the images makes image interpretation very difficult. Problems in dealing with perspective changes and the wide range of object scale and image resolution were encountered. In addition, there are many sources of geometric and radiometric variability which confound attempts at object detection. Finally, there has been a lack of adequate models relating physical principles to object appearance in images. Work on developing expert systems for image understanding and for cartographic feature extraction is motivated by the hope that these problems may be successfully handled with the use of knowledge-based approaches.

Expert systems and knowledge-based systems are terms which refer to

systems in which problem domain knowledge, specific data, and control strategies to manipulate the data and the knowledge, are not embedded, implicitly, as part of the program code but more formally identified and organized in a structured manner. This paper discusses some basic issues in the design of expert or knowledge-based systems for understanding remotely sensed images, with particular reference to cartographic feature extraction problems. A well-known example of an image understanding system is ACRONYM developed by Brooks (1983). An examination of the strengths and limitations of this notable system points to the additional capabilities that will need to be provided in order to succeed in developing an expert system for understanding remotely sensed images and extracting cartographic features.

Section 2 presents an overview of knowledge for feature extraction in remotely sensed imagery, including sensor, atmospheric, and terrestrial factors inducing variability in the sensed image; use of knowledge about objects, image processing, object recognition and modelling; use of map and elevation data; characteristics distinguishing cartographic feature extraction from other types of image processing tasks; and the role of high-level languages in entering expert knowledge into an expert system.

The complexity of understanding remotely sensed images has led to interest in fusion of information from various sources such as high resolution monochromatic imagery, radar, infrared, and knowledge bases. Section 3 mentions ways in which different types of inputs can be used together with image interpretation and

indicates problems arising from the varying resolution in different types of sensor imagery. The presence of various inputs and various expert modules such as roads, water, texture and spectral analysis experts gives rise to the presence of several estimates as to the likelihood of various objects being present in a given part of the image. Several interesting procedures for treating this problem of combining estimates and evidence have been proposed in the literature. In Section 3 we briefly discuss two approaches, including one called Consensus Theory which has not hitherto received attention in the literature on expert systems.

Sections 4 and 5 discuss the architecture and control considerations we deem important for expert systems for understanding remotely sensed images and extracting cartographic features. Section 4 discusses why distributed problem solving and distributed architectures are suitable for this application and discusses the ACRONYM image understanding system in this context. Section 5 describes a possible control mechanism for handling a distributed expert system which appears particularly relevant in this context. This includes a top-down and bottom-up parallel search procedure for searching AND/OR graph models and its parallel, non-directional implementation. We also discuss the use of this approach in dynamic modelling and dynamic control of distributed expert systems.

Section 6 mentions some of the open problems which need to be addressed in order to realize expert systems for understanding remotely sensed images and Section 7 presents concluding remarks.

2.0 KNOWLEDGE FOR FEATURE EXTRACTION IN REMOTELY SENSED IMAGERY

The successful interpretation of remotely sensed images requires many types of knowledge. Many types of entities can appear in an image and associated with each is a large collection of properties. Expert systems that interpret images need knowledge about image processing, image formation, object recognition and object modelling. Expert systems that need different types of knowledge should be able to handle multiple representations. There is a great deal of knowledge particular to the remote sensing field. In this section, we discuss some of the types of knowledge required for understanding remotely sensed imagery and extracting cartographic features.

2.1 REMOTE SENSING

First, knowledge about the external factors that induce variability in aerial imagery should be included. Automated or semi-automated recognition processes must be able to adjust to or to account for variations that can be expected to appear in remote sensing and that are not caused by changes in the objects themselves. This knowledge derives from current sensor models, and models for geometric and radiometric distortion introduced in the image. We briefly mention some of these factors. More extensive discussions of this subject can be found in remote sensing texts such as (Lillesand 1979) and (Swain 1978).

Remote sensing is a method of gathering information about an object without any direct contact with the object. Most of the sensors used in imaging remote sensing applications

are designed to respond to different parts of the electromagnetic spectrum. Some sensors such as multi-spectral scanners are passive where the sensor records ambient energy caused by solar and terrestrial radiation. Some sensors may be active as in the case of radar systems where they provide the radiation which is reflected off the object.

The sun radiates mainly in the visible and infrared (IR) parts of the electromagnetic spectrum and the earth radiates mainly in the thermal infrared (TIR) region. Not all of this radiation can be used by the remote sensor since particles and water vapor in the earth's atmosphere both scatter and absorb some of it. The overall effect of scattering is to decrease the amount of energy reaching the surface and to create "airlight" or background haze. The airlight can then enter the remote sensor and cause deterioration in the quality of the imagery. The effect of absorption is to decrease the amount of energy reaching the ground.

As a result of the scattering and absorption effects, only some regions of the spectrum reach the ground. This results in "atmospheric windows" (i.e., bands of the spectrum where transmission is approximately 70-100%) which are useful for remote sensing. These windows occur in the visible, near IR, middle IR, and thermal IR regions. The electromagnetic radiation which reaches the ground can then be reflected or thermally emitted by objects and this energy is then sensed by the remote sensors.

Models approximating the solar and terrestrial radiation, and the scattering and absorption effects represent some of the knowledge that should be included in any expert system

for remotely sensed imagery. In addition, temporary conditions such as haze, rain, and snow will also affect the appearance of objects. Knowledge of these effects should also be available to the expert system.

There are many types of remote sensors, e.g., photographic aerial cameras, multispectral line scanners (MSS), thermal infrared scanners (TIR), and side-looking airborne radars (SLAR). Each of these types of scanners and the distortion they introduce may be approximated by models. This also represents knowledge that must be incorporated in cartographic expert systems which interpret remotely sensed images.

The two basic types of distortion are radiometric distortion and geometric distortion. Radiometric distortion occurs when the intensity of the radiation received by the sensor undergoes changes through absorption and reflection processes as discussed above. Different conditions, such as the amount of dust or water vapor in the atmosphere, changes in the sun-object-sensor angle, and the type of surface of the objects, can greatly affect the appearance of the objects. In addition, the response of each detector in a sensor is different from any other detector. All these effects cause radiometric distortions.

In addition to these radiometric effects, the images are affected by geometric distortions which can also greatly change the appearance of the objects. For example in some scanner imagery, straight roads will become "S" shaped. The sensor geometry introduces geometric distortion which changes the physical appearance of an object. Even when sensor-based geometric distortions are partly corrected, other geometric distortions present can not always be removed.

The basic types of geometric distortions common to all remote sensors are:

- 1) Distortions in the image introduced by the topography, i.e., by the varying elevation of the terrain.
- 2) Distortions caused by the sensing mechanism.
- 3) Distortions caused by the recording mechanism.
- 4) Distortions caused by a non-ideal flight path of the aircraft carrying this sensor.

Some types of aerial images have additional distortions: at high altitudes both the curvature of the earth and its motion become noticeable. Clearly an expert system for cartographic feature extraction should have access to models of complex phenomena, such as the above types of distortion, not all of which are fully developed at present.

2.2 OBJECTS

Knowledge about the objects which are likely to be encountered must be included in the expert system. This information may include shape, structural, material composition and surface properties of the objects. Three dimensional models of the object may be required. Relational properties of the object may also be necessary. For example, the knowledge that a bridge joins two bodies of land across a body of water or of land of lower elevation can be of great use in locating bridges.

Further properties of the object could describe their temporal variation or their interaction with the environment. Crops change radically in appearance throughout the year. Soil reflectance can change drastically with variations in the moisture content. Functional properties of the objects can be greatly useful in recognition for transportation and thus they are likely to be found near roads.

The physical structure of objects can be used to predict their appearance in various types of remotely sensed images. This knowledge should be provided and should include methods for determining radiometric response characteristics. Principles of geometric reasoning should also be included. An example of geometric reasoning is the effect of perspective in viewing three-dimensional shapes.

2.3 IMAGE PROCESSING

Image processing knowledge must obviously be represented in the expert system. This knowledge should include description of the wide range of image processing techniques, their uses, and the information required to use them. Information about the image formation and acquisition processes should be available. In some imagery the original data may not lie on a rectangular grid, so various interpolation processes may be necessary to transform the data. This is true for several widely used data sources such as Landsat and Seasat data. The transformations will affect the accuracy of the image, which may be of great importance if high precision measurements are required. In general, the expert system should know the image scale and the characteristics of the geometric and radiometric distortion

resulting from the sensor. Since a wide range of object sizes are present in remotely sensed images, knowledge of how to adapt the image processing techniques to objects changing widely in size should be included. Varying image resolutions give rise to similar problems and must be dealt with.

2.4 OBJECT RECOGNITION AND MODELLING

An image understanding system should include knowledge about object recognition and object modelling. Object recognition is the process of recognizing an instance of an object given a model of it. Object modelling is the process of separating an object from the background given generic descriptions of relations between object components. An example of the object modelling process is Waltz's work on corner labelling to find objects in an image (Waltz 1975).

2.5 CARTOGRAPHIC FEATURE EXTRACTION

Cartographic feature detection is not a well-defined process. The goals of different photo-interpreters, photogrammetrists, and cartographic researchers can be completely different, since the desired features and their characteristics may be totally different. Thus, any cartographic feature extraction system is going to have to be flexible in order for it to be generally useful.

The desirability of map-guided image interpretation is now recognized by many researchers in cartographic image processing, though there is considerable variation in their approaches. Feature verification using a map is constrained enough that

simple procedures such as template matching may be sufficient. But this simplicity may come only after registration of the acquired image with a map, and such registration may itself require complex processing. Maps can be used to limit the type of features that can be present in various parts of the image. This reduces the amount of object detection computation. For example, it would not be useful to apply car detectors in a region identified as a lake, but boat detectors or island detectors would be useful.

Elevation information from a database or derived from stereo pairs, has not been widely used in remote sensing. This has been due, in part, to the lack of adequate elevation data. As noted earlier elevation information can be useful as an additional feature for cartographic feature extraction.

Since edge detection is commonly performed on images in order to object primitives for object recognition it is desirable to have evidence for the correctness of an edge. Edges which coincide with significant differences in elevation may be useful in rapidly locating significant region boundaries. This may be especially useful in locating and classifying man-made objects such as buildings. In this type of classification preliminary edge detection may be done using only edges defined by elevation differences.

Elevation data can be useful in classifying some natural features such as water. Artificial edges may appear in water due to shadows turbulences, etc. If elevation data indicates no change in height at artificial edges, then it may be possible to label these edges as caused by shadows.

While cartographic feature extraction is a subfield of image analysis, there are some differences between computerized cartographic feature extraction and the rest of image analysis (Nagao and Matsuyama, 1980). An advantage of cartography is the assumption that most objects in aerial imagery can be treated as 2-D. Being able to ignore the 3-D aspect of objects greatly simplifies some modelling problems but also means that an object's three-dimensional shape cannot be used to discriminate it from other objects.

Unfortunately, cartography has other problems not shared with the rest of image analysis. First, the images tend to be much larger and there is the problem of fusion discussed in a later section. Second, the information content in one image can be extremely large. For example, a single image can contain part of a city, some related suburbs, and surrounding farms and terrain, yet the resolution may be good enough to resolve individual buildings. This brings up the third problem: the wide variability in the size of objects. Some objects may be large, such as a forest, while other objects may be relatively small, such as vehicles. Yet all of these objects can be of interest to a photointerpreter. The fourth problem relates to the factors discussed in Section 2.1 - that external concerns such as the atmosphere can cause changes in the object's appearance.

2.6 ENTERING EXPERT KNOWLEDGE

The importance of context and relations among cartographic features makes the use of high-level knowledge a necessity. The system should use expert knowledge to process the results of the low-level image pre-

processing. In order for the expert system to be effectively used, a method for easily entering expert knowledge is needed. The cartographic expert should not have to be a computer expert nor should he have to be well informed about the specifics of the system. This forces the structure of the system to be both modular and easily modified. This means a high-level representation language is needed to represent this expert knowledge (Reggia 1981, Lambird, 1981).

The language will have to be able to describe complex relationships since cartographic features are complex. In addition, the language must provide structures for representing image features such as regions and provide means for manipulating these features. System-supplied functions appropriate to image features such as LENGTH and AREA will be needed as part of the descriptions of the image features. The structures for representing cartographic features and their allowed relationships will have to be compatible with the method for representing maps.

3.0 FUSION OF INFORMATION

The complexity of many image processing tasks has generated interest in the possibility of combining information from a variety of sources. Radar, multispectral and infrared imagery are but a few of the many sources of pictorial information available. Elevation matrices, maps and expert rule systems provide additional knowledge useful for image interpretation. While the tremendous amount of information available from these sources offers hope for considerable improvement in the

reliability of scene analysis, many obstacles must be overcome. We describe here, some of the major problems encountered in performing this integration and indicate some directions of research which may be useful in surmounting these problems.

Complex search problems abound in many scene analysis tasks. The linking of short edge segments to form curves and the splitting and merging of regions to find more reliable regions are common examples of large image processing search problems. Multiple sources of information can, in some cases, be used to reduce search complexity, while in other cases it can greatly increase complexity. The detection of roads is a promising application of fusion of information. Line or edge detectors in high resolution monochromatic imagery can be used to locate prospective sites for roads. Elevation information obtained from stereo imagery imposes constraints on possible road locations. Multispectral imagery can be used to check for road surface materials. Rules limiting the curving of roads in some locales may be used to further limit possible road detections.

The weighting of information from disparate sources presents many problems. Errors frequently arise in the various knowledge sources. Edge detectors often yield spurious edges. Stereocompilation methods for determining elevation are unreliable in many types of terrain, such as lakes. In light of these inaccuracies any system for reasoning about images should be able to handle the contradictory evidence arising in the search process.

A second problem arising in pictorial information fusing is the handling of images at different resolutions. The resolution of aerial monochromatic

imagery is generally much higher than that of Landsat multispectral imagery. Aerial thermal imagery tends to be at very low resolution. Variation in image distortion due to differences in sensor characteristics creates further problems in mixing information. The registration of images from different sensors at different resolutions requires some type of feature based matching (Stockman, et al, 1981).

3.1 CONSENSUS THEORY

The fusion of information from various sensors can lead to problems in combining beliefs in hypotheses generated by the various images or databases. For example, each of several images obtained using different sensors may indicate the presence of a lineal in approximately the same location. Due to uncertainties in lineal feature extraction, a lineal expert may assign to each sensed image a set of weights indicating belief in various possible locations for the lineal. In some cases, knowledge of the regions bordering the lineal will be sufficient to select one sensed image as being all important for the lineal detection. For example, the lineal may be a river boundary and it may be known that water boundaries are extremely reliable in infrared images. More generally however, it will be necessary to combine the distributions in some way. In addition to the distributions on lineal location, each image may have assigned to it a binomial distribution on the presence or absence of the lineal.

The work of Bayesian subjective probabilists has focussed on the problem of optimal decision making once one has combined the various distributions into a single distribution. A number of schemes for assigning

scores to the various experts and combining these scores to aggregate the experts distributions have been studied. Various approaches to this task, referred to as finding a consensus probability distribution are reviewed in Winkler (1968) and Hogarth (1975). It should be noted that an alternate approach to decision making problems is to have the experts collectively make the decision rather than have a single decision maker combine the experts' distributions. This type of group decision making is discussed in the economics and psychology literature (Fishburn 1973) and (Hoffman 1965). Two types of procedures for combining experts' distributions for use by a single decision maker are often discussed. In the first method, called the no-interaction method, each expert, without knowledge of the work of the other experts, develops a probability distribution and the decision maker combines them. In the second approach, called the group-interaction approach, the experts are allowed to confer in coming up with a distribution representing group opinion.

To understand more fully the potential applications of consensus theory in expert systems, it is necessary to examine belief manipulation methods currently under investigation in connection with expert systems. We first note that classical Bayesian theory does not treat the problem of combining probability distributions, since the experts' distributions need not bear any particular relation to the underlying distribution of events, nor to each other.

3.2 EVIDENCE THEORY

The theory of evidence developed by Arthur Dempster (Shafer 1976) is

currently under study (Barnett 1981) as a means of combining beliefs from various experts. In this theory the notions of probability density functions and distributions are generalized to reduce the problem of overcommitment required of experts in probability density estimation. Roughly speaking, specifying a probability density function requires one to give a probability mass for every point in one's sample space and all further probability calculations are built out of this specification. In the theory of evidence, it is possible to assign a belief to a subset of a sample space where this belief is now merely obtained by integrating the belief function over the points in the subset. Thus given an event, some of our beliefs for this event may be attributed to our belief in the various subevents whose union is this event, while part of our belief may be attached to the event as a whole without any opinion as to how it is distributed over subevents.

The need for the above generalization arises frequently in image processing (Garvey, Lowrance, Fischler 1981). However, Dempster's theory goes further and specifies a rule for combining the generalized densities to form new generalized densities. The rule is simple, intuitively appealing and specializes to a rule which had previously been used in the expert system MYCIN (Shortliffe 1976) and others. This combination rule, when specialized to probability distributions is a particular instance of a no-interaction consensus rule. We can characterize the difference between consensus theory and the theory of evidence as follows. The theory of evidence generalized the notions of probability theory and uses a particular rule for combining generalized probability densities while consensus theory studies a variety of

approaches to combining ordinary probability distributions to form a new distribution.

4.0 DISTRIBUTED PROBLEM SOLVING

The knowledge sources needed for understanding remotely sensed images are so varied and the data is so voluminous that a distributed problem solving (DPS) approach appears most promising. In this section we discuss the nature of distributed problem solving and suggest a possible architecture for distributed cartographic expert systems.

4.1 NATURE OF DPS

In distributed problem solving (DPS) a complex problem is subdivided into a set of distinct subproblems which are easier to solve. Each subproblem can then be assigned to a processor for solution. DPS has several advantages (Chandrasekaran 1981). Each processor need only solve a more limited problem. The total input to the limited problem should be correspondingly smaller. Distributing the problem among several processors allows parallel processing to take place. In cases which require real-time processing, parallel processing may be absolutely necessary in order to accomplish the tasks. If multiple processors are used then parts of the system can fail or degrade but still allow partial results to be obtained, or at least the cause of failure to be determined. Finally, distributed problem solving allows a modular structure which can be easier to expand or be more adaptable to change, if properly constructed.

There are basically two architectures for distributed problem solving:

network and hierarchy. In a network of processors, each processor can communicate directly with any other processor. While this architecture facilitates exchange of information, it requires that all processors must have the information processing capability of the processor with the largest information load. Thus this architecture involves a trade-off of increased ease of inter-processor communication against a requirement that all processors must be complex enough to handle the greatest information load.

In a hierarchy of processors, the lines of direct communication are limited to be between only those processors which are directly connected. Thus, some processors can not directly communicate, but must go through intermediary processors. In this case, most processors need no longer be capable of handling the greater information load. However, since information may now be passed through intermediary processors, "filtering" or "biasing" of the information will occur. This biasing may be good or bad, depending on the application. For example, suppose a processor inquires through the hierarchy, if a detected feature on an unrectified line scanner image is a straight road. As the inquiry is passed through the hierarchy, the line scanner expert would appropriately change this inquiry since straight lines are distorted into various shaped curves depending on the orientation of the line with respect to the line scanner. In this example, the biasing or filtering of the information is necessary. However, the hierarchical architecture requires that the hierarchical structure be constructed. Thus this architecture involves a trade-off between decreased ease of inter-processor communication with the added complexities of information

filtering, and a lessening of the requirement that all processors must be complex enough to handle the greatest information load.

In the past, most expert systems have distributed the knowledge (for example, into discrete rules, frames, etc.) but have kept the control or processing burden on a highly centralized controller. For applications with a very high information and processing load, such as all image understanding expert systems, the burden rapidly taxes the capabilities of most systems. In order to construct useful image understanding expert systems, the processing burden will have to be distributed. This means new expert systems architectures need to be developed. Later parts of this paper describe a general distributed architecture for expert systems and a method for allowing parallel processing control.

4.2 DISTRIBUTED EXPERT SYSTEMS

Chandrasekaran (1983) has suggested that expert systems can be organized as a "cooperating community of specialists". In this case, the knowledge is divided among a set of structures each of which utilizes the most appropriate type of problem solving method. Each structure can then be decomposed into a hierarchy of specialists which share the same type of problem-solving method. In this architecture, knowledge is not separated from the control mechanism but is embedded in it. Thus, the problem-solving is distributed throughout the expert system.

Each specialist contains its own knowledge base and corresponding inference mechanism. Usually specialists high in the hierarchy are more general, while specialists lower in the hierarchy are more specific. For example in a cartographic expert system a higher-level specialist could be a specialist in interpreting urban areas, while a lower-level specialist could be a specialist in the recognition of automobiles. Communication between specialists can occur readily along the lines of the hierarchy. Communication between specialists not directly connected by the hierarchy can be accomplished either through the hierarchy or through an external blackboard. The blackboard contains the status of the system, i.e., what specialists have been explored and their status.

The distributed expert system has several advantages. First it allows different types of knowledge representation and the appropriate problem-solving methods to be included in one system. This advantage is extremely important for cartographic expert systems, since there is a large amount of very different kinds of knowledge needed for the interpretation of remotely sensed images. The requirement of only one knowledge representation and corresponding problem solving method is too restrictive. This subject is explored in more detail in the next section. Much of the control and "focus of attention" problems are alleviated since control can only pass the lines of the hierarchy. The distributed architecture does have disadvantages. The domain knowledge must be carefully structured in a hierarchy. In addition, in a large complex system communication through the blackboard may be difficult to implement efficiently.

4.3 ACRONYM

ACRONYM (Brooks 1983) is an image understanding expert system that does symbolic reasoning on two-dimensional images using three-dimensional models. It incorporates three separate expert systems each of which have their own knowledge representation and type of reasoning. The three expert systems cooperate to interpret the image. Very briefly, the expert systems are: (1) A "prediction" system that uses the three-dimensional models to predict geometrically invariant features to look for in the image. This system uses geometrical reasoning. (2) A "description" system that uses the images to get descriptions of possible image features. This system uses image formation reasoning. (3) The third system is an "interpretation" system that uses the descriptions from the second system to find constraints and check the consistency of the results. This system uses graph matching to perform its reasoning. The three systems are iterated (prediction-description-interpretation) in order to get a finer and finer detail interpretation of the image.

ACRONYM includes some knowledge about object recognition object modelling, image formation, sensor model, and illumination model. However, most of this knowledge is minimal. ACRONYM has a sophisticated modelling system for three-dimensional objects, but the objects must be rigid or be composed of rigid components. In addition, only limited object relations and properties are handled by ACRONYM. Thus, while ACRONYM represents a significant step in the inclusion of knowledge, many additional features must be added in order to incorporate knowledge of the type described in Section 2.

The three expert systems in ACRONYM are each rule-based production systems which do not explicitly relate the knowledge in a hierarchy of specialists and must be separately controlled. The focus of attention problem mentioned above could lead to difficulties. As reported in the literature, ACRONYM has only been tested on a limited amount of imagery for a small set of objects. The control mechanisms used in ACRONYM are unlikely to be able to handle interpretation of complex images containing a large set of complex objects. For the cartographic feature extraction problem, most general control structures should be considered. In the next section, we briefly describe a control approach which we have been investigating because we feel it is more suited to this domain.

5.0 CONTROL

Using an expert system to extract cartographic features from an image may be thought of as a problem in search. The requirement that the search be distributed among a set of processors gives rise to the need for a parallel search algorithm. This algorithm should require communication of small amounts of information at infrequent intervals since each expert module will require a fairly sophisticated processor and the intimate communication among many such processors is time consuming and prone to failure.

Further requirements are imposed on the search method by the cartographic problem. Due to the large amount of information present in images, expectations as to the likely contents of the scene should be used to direct the search for primitive features.

These expectations should be computed during the course of the scene analysis. For example, the knowledge that a large body of water is likely to be present in an image may trigger the application of certain texture measures to locate the body of water. On the other hand, there are many features which may occur in a given image even though there is no a priori means of assessing the likelihood of their occurrence. In such situations, primitive image processing operators such as edge or texture detectors may be applied to the image to uncover the possible presence of higher level structures such as buildings. This "bottom-up" search can be very time-consuming since it leads to a large number of hypotheses, most of which are wrong. As soon as the presence of such a higher level structure is suspected, it may be reasonable to resume a top-down approach, where high-level knowledge can be used to guide the search. A search algorithm used to control the cartographic expert system should be capable of moving between top-down and bottom-up modes of search depending on the characteristics of the data found so far.

The next two sections describe a parallel search procedure capable of both top-down and bottom-up modes of search. This procedure was originally developed (Stockman and Kanal, 1983) to guide search in the analysis of medical waveforms using methods from syntactic pattern recognition. The underlying search procedure, known as SSS*, has been the subject of considerable analysis (Stockman 1979), (Roizen 1983), and (Campbell 1981). The waveform analysis package, known as WAPSYS, which contains SSS* has proven to be a fast, flexible means for the analysis of several types of medical waveforms (Stockman and Kanal, 1983), (Xiong,

et al., 1984). The third section discusses two aspects of the dynamic control that will be needed in the cartographic expert system.

5.1 A TOP-DOWN AND BOTTOM-UP, PARALLEL SEARCH PROCEDURE

The SSS* algorithm is a best-first state space search procedure developed for application in structural pattern recognition. This algorithm was designed to provide flexibility in search by allowing both model-directed and data-directed search. In particular, the following design criteria were used:

- 1) Ambiguous interpretations should be allowed and developed in a best-first manner.
- 2) A priori knowledge of the problem domain should be available to make hypotheses about the data for subsequent verification.
- 3) Key events in the data, as detected by low-level interrogation, should be capable of triggering a search for higher level structures in the data.
- 4) The order of application (2) and (3) should be determined dynamically based on some optimality criterion.

In addition to the above features, SSS* is readily adaptable to distributed problem solving.

SSS* uses a problem reduction representation (PRR), AND/OR graph, or a grammar database. Structures of interest are decomposed in terms of alternative (OR) and component (AND) substructures. Structures which

cannot be further decomposed are regarded as primitives. In the cartographic context, the AND/OR graph consists of modules describing features such as roads, buildings, forests or towns or methods for recognizing cartographic features. The descriptive components may include a variety of descriptors, such as the geometric layout of an object, texture properties, and spectral features. Primitives are detected by low-level image processing operations or by interrogating a human user of the system. The large number of primitive detectors applicable to a given image necessitates the use of a good control strategy.

SSS* controls the search by matching the PRR model to data. The algorithm terminates, if possible, with a description of the structure in question. This description takes the form of a state tree for the structure. Components of the tree are labelled and parameter values are present when appropriate.

A key feature in the efficiency of the search algorithm is the ordering of components based on the expected ease of detection. This ordering may be based on experimental studies or subjective assessment. By examining the components using this ordering, inappropriate branches of the AND/OR graph can be rapidly eliminated. For example, the component which is easiest to detect may be a large rectangular structure. This component was listed as best in the ordering since it was the easiest to detect. If this structure is not found, then the more time-consuming search for one of the other components need not be performed.

Matching of the PRR to data is accomplished by applying a sequence

of operators which generate a sequence of states representing partial parse trees. Both top-down and bottom-up operators may be applied, depending on whether a structural goal has been set or a structure has been recognized. Partial parse trees are stored in a linear encoding and rated as to the quality of the match with the data. As alternative OR goal structures are generated, WAPSYS creates competing trees. This parallel development of competing trees allows discovery of multiple solutions to a problem.

WAPSYS uses the A* search algorithm defined in Nilsson (1980) to control generation of the states. A global database known as the State Space Representation (SSR) contains an encoding of the states which have been generated. Initially, the only states in the SSR are structural goals for top-down analysis and primitive structural goals for bottom-up analysis. Each state is assigned a merit which is an estimate of how far the state is from a goal state. The A* algorithm expands the state which has the highest merit value and places the expanded state in the SSR. Currently WAPSYS uses the minimum value of any recognized primitive structure in a state as the value of the merit. This approach of defining the quality of recognition of an object as being the merit of the most poorly detected part leads to some difficulties in search. One moderately bad but acceptable evaluation will greatly reduce the chance of the state being expanded until other states which, on the average work, have been expanded. Various schemes to overcome this limitation have been devised using some type of average error, but they are not guaranteed to arrive at an optimal solution.

The operators used in the A* state space search provide the ability to incorporate both top-down and bottom-up search. These operators, are described in more detail in (Stockman 1983). Recognized structures may have associated attributes. These attributes may contain information such as the physical location of the recognized structure. These attributes can be used to constrain later parts of the search.

The primary application of WAPSYS has been to the analysis of waveforms, though the paradigm has a much broader range of applications. Through its ability to combine top-down and bottom-up search, and its ability to develop competing solutions in parallel, WAPSYS promises to provide a flexible control structure for the problem of distributed expert system processing.

5.2 PARALLEL NON-SEQUENTIAL SEARCH

A parallel implementation of the SSS* search algorithm has been developed in the context of Branch and Bound (B&B) procedures (Kanal 1981, Kumar 1984). The search space is partitioned and each part is searched in a depth-first search. Each time alternate structural models arise in the AND/OR graph search, the AND/OR subtrees corresponding to the alternatives, can be sent to separate processors. Using this approach, the procedure can be operated as a set of processors working independently and asynchronously. Only the merit of the best partial parse tree encountered in the search so far need to be communicated among the processors.

We now briefly describe the general framework for parallel search and a

parallel implementation of SSS* developed in Kanal (1981) and Kumar (1984). The basic approach to B problems is to decompose the search space into disjoint areas and search each area in a depth-first fashion. These searches can be performed concurrently. Each time a processor encounters a better solution, it informs all other processes in the system of the merit. Whenever a partial solution is encountered which cannot be better than the current global optimum, that solution is discarded. Since a processor only uses shared information to decide if it should give up a partial solution and examine another part of the search space, it never has to wait for input from another processor. This feature is highly desirable if the algorithm is to be implemented on a loosely coupled architecture on which interprocessor communication time is slow. Evaluation of the speedup achievable with this parallel algorithm is difficult to evaluate on theoretical grounds since it depends largely on how well the search space can be divided into areas that require approximately equal search time.

A solution tree T of an AND/OR tree G , is usually defined as a sub-tree such that (a) the root node of G is the root node of T and (b) if a non-terminal node of G is in T , then all of its immediate successors are in T if they are of type AND and exactly one of its immediate successors is in T if they are of type OR. Complementary to this "AND" solution tree formulation, an "OR" solution tree may be defined as follows (Kanal 1981): (a) the root node of G is the root node of solution tree T , and (b) if a non-terminal node of G is in T then all of its immediate successors are in T if they are of type OR and exactly one of its immediate successors is in T if they are of type AND.

The parallel implementation of the B&B algorithm requires depth first search in the AND solution tree space and best first search in the OR tree search space. Unfortunately, sequential SSS* does the reverse of this. To overcome this problem, a dual version of SSS* has been formulated, in which partial OR solution trees are searched in best first order. This approach called dual-SS* is a depth first search in the AND solution tree space. Dual-SS* keeps track of the best complete AND-solution tree which has a merit lower than the current best solution is eliminated. Furthermore, the current best solution tree is replaced only when a solution tree with higher merit is encountered. Thus the dual-SS* algorithm can be used for the search in the parallel SSS* implementation.

Although the distributed architecture described in this section seems most suited to cartographic applications, much research needs to be done and many problems to be solved before a true cartographic expert system can be realized.

5.3 DYNAMIC CONTROL

The advent of distributed expert systems naturally gave rise to the possibility of using a loosely coupled network of processors. While the assignment of different expert modules to different processors may appear to be a natural means for assigning software to processors, this can lead to serious difficulties. Expert modules differ considerably in their system resource requirements on both a static and dynamic level. From the static point of view, the amount of code occupied by different expert modules will vary. In addition, the a priori expected usage of these modules

will also vary. On the dynamic side, the frequency of usage of certain modules will depend heavily on the image being processed at a given time. Ideally, run time assessments of images under analysis should be performed to aid in the decision as to how to distribute the modules. A simpler, but still nontrivial approach is to assume the modules are assigned to fixed processors and focus on the problem of controlling the flow of information.

A second type of dynamic control lies in the run-time construction of the AND/OR graphs. As discussed in earlier sections, the distributed problem-solving system consists of a hierarchy of specialists. When a specialist is invoked, it must decide which if any of its sub-specialists it must call upon to help solve its problem. In cartography, the applications are too complex to enumerate all the possible combinations of specialists that would be needed to solve all possible problems. For example, towns have such an enormous number of variations that it is not feasible to have an explicit AND/OR graph to capture these possibilities. As an alternative, the structure of the AND/OR graph can be dynamically constructed during the search process. The selection of specialists will be based on the static structure of the underlying tree of the expert system and the data observed so far in the search.

As an example, let one of the specialists in the cartographic expert system be a specialist on the recognition of urban areas. Since there are many different types of urban areas (for example, heavy industry, light industry, suburban, etc.), there will probably be sub-specialists which specialize in the recognition of these different types.

Thus, when the urban area specialist is invoked, it may have enough information to realize the area is either a light industrial area or a suburban community. In this case, the dynamic AND/OR graph need only include those two-sub-specialists in an OR configuration. Depending on the data observed in the search so far, the light industrial urban area specialist may invoke experts to seek railroads, waterways, large building complexes and evidence of pollution. The suburban specialist may institute a search for road networks and large numbers of small buildings in highly structured patterns. These two specialists are invoking their sub-specialists in an AND configuration.

6.0 SOME PROBLEMS TO BE SOLVED

In this section, we present several problems that are illustrative of the type of problems that need to be solved before practical remote sensing expert systems can be effectively developed and implemented.

A fundamental problem is the resolution of contradictory information resulting from the use of several types of sensors. In the early stages of processing, the different sensed images may be handled independently. Radar, multispectral, and infrared imagery are but a few of the sources of pictorial information available. Elevation matrices and maps provide additional knowledge useful for image interpretation. While the tremendous amount of information available from these sources offers hope of considerable improvement in the reliability of scene analysis, many obstacles must be overcome.

Complex problems in interpreting information from multiple sensors are

common in scene analysis. For example, long narrow features such as roads have been used to compute the transformation needed to register (align) two images of the same area. Preliminary feature analysis may indicate the presence of roads in the two images, but due to various types of distortion, the shapes of the roads may vary enough to make accurate registration difficult. We have encountered this problem in work on registering radar and optical images. A simple weighting of the beliefs in the reliability of the two sources of information can lead to estimated road positions which are seriously inconsistent with both sources. In this type of problem, the expert system is required to develop geometric descriptions satisfying a variety of quantitative and qualitative constraints.

Once an image is registered to other images, maps, and elevation matrices, search for other roads can make use of the multiple information sources. Line or edge detectors in high resolution monochromatic imagery can be used to locate prospective sites for roads. Elevation information obtained from stereo imagery imposes further constraints on possible road locations. Multispectral imagery can be used to check for road surface materials. Rules limiting the curving of roads in some locales may be used to further limit possible road detections.

A major problem that is inherent in developing any distributed expert system that would use parallel processing is the problem of assignment of expert modules to particular processors. A successful solution to this problem will require procedures for predicting the relevance of expert modules and the complexity of their search problems during the analysis of a scene. Such

prediction procedures could initially be based on some simple measures of search complexity such as some measures of information content in an image (such as average edge density or texture complexity in sample areas). This problem is made even more difficult by the dynamic structuring of the AND/OR graph. This problem of predicting the relevance of the expert modules for the purpose of processor allocation is a virtually unexplored area.

7.0 CONCLUDING REMARKS

We have described the complex types of knowledge that will have to be available to a remote sensing expert system. We have described the distributed architecture and a control structure based upon a parallel non-directional search algorithm which we consider highly suited to this application. We have also mentioned a number of problems which still need to be solved before practical systems can be realized. We have also briefly mentioned information fusion and methodologies for processing contradictory or ambiguous evidence which are important, challenging problems in themselves. Many aspects of expert systems for remotely sensed images have not been touched on in this paper. Some of these, e.g., the role of low-level statistical processing and classification integrated with the higher level expert system are described in our previous reports (Lambird 1981). We are continuing to explore the ideas presented in this paper and implementing some of them in some of our current projects.

REFERENCES

1. BARNETT, J.A., "Computational Methods for a Mathematical Theory of Evidence", USC/Information Sciences Institute, Proc. 7th Int'l. Joint Conf. on A.I., Vancouver, B.C., Aug. 1981
2. BROOKS, R.A., "Model-Based Three Dimensional Interpretations of Two-Dimensional Images", IEEE Trans. on PAMI, 5, 140-150, March 1983.
3. CAMPBELL, M., "Algorithms for the Parallel Search of Game Trees", Tech. Report 81-8, Dept. Comp. Sci., University of Alberta, Edmonton, 1981.
4. CHANDRASEKARAN, B., "Natural and Social System Metaphors for Distributed Problem Solving: Introduction to the Issue", IEEE Trans. on SMC 11, 1-4, January 1981.
5. CHANDRASEKARAN, B., "Expert Systems" Matching Techniques to Tasks", New York University Symposium on Art. Int. Applications for Business, May 18-20, 1983.
6. FISHBURN, P.C., The Theory of Social Choice, Princeton University Press, Princeton, NJ, 1973.
7. GARVEY, T.D., Lowrance, J.D. and Fischler, M.A., "An Inference Technique for Integrating Knowledge from Disparate Sources", Proc. 7th Int'l. Joint Conf. on A.I., Vancouver, B.C., 319-325, August 1981.
8. HOFFMAN, L.R., "Group Problem Solving", Advances in Experimental Social Psychology II, L. Berkowitz (ed.), Academic Press, N.Y., 1965.
9. HOGARTH, R.M., "Methods for Aggregating Opinions", Proc. 5th Research Conference on Subjective Probability, Utility and Decision Making, Darmstadt, 1975.
10. KANAL, L.N. and Kumar, V., "Parallel Implementations of a Structural Analysis Algorithm", Proc. IEEE Comp. Soc. Conf. Pattern Recognition and Image Processing, Dallas, 452-458, 1981.
11. KUMAR, V. and Kanal, L., "Parallel Branch and Bound Formulations for AND/OR Tree Search", to appear in IEEE Trans. on PAMI, 1984. LAMBIRD, B., Lavine, D., Stockman, G., Hayes, K., and Kanal, L., "Study of Digital Image Matching of Dissimilar Images", ETL0248, USAETL, Fort Belvoir, VA, January 1981.
12. LAMBIRD, B., Lavine, D., and Kanal, L., "Interactive Knowledge-Based Cartographic Feature Extraction, ETL-0273, USAETL, Fort Belvoir, VA, October 1981.
13. LILLESAND, T. and Kiefer, R., Remote Sensing and image Interpretation, John Wiley and Sons, 1979.
14. NAGAO, M. and Matsuyama, T., A Structural Analysis of Complex Aerial Photographs, Plenum Press, New York, 1980.
15. NILSSON, N., Problem-Solving Methods in Artificial Intelligence, Tioga Publishing Co., Palo Alto, CA, 1980.

16. REGGIA, J.A., "Knowledge-Based Decision Support Systems: Development Through KMS", Ph.D. Theses, University of Maryland, 1981.
17. ROIZEN, I. and Pearl, J., "A Minimax Algorithm Better than AlphaBeta? Yes and No", Art. Int. 21, 199-220, March 1983.
18. SHAFER, G., A Mathematical Theory of Evidence, Princeton University Press, Princeton, NJ, 1976.
19. SHORTLIFFE, E.H., "Computer Based Medical Consultations: MYCIN" Artificial Intelligence Series, Volume 2, American Elsevier, Inc., NY, 1976.
20. STOCKMAN, G.C., "A Minimax Algorithm Better than Alpha-Beta?", Art. Int. 12, 179-196, 1979.
21. STOCKMAN, G.C., Lambird, B., Lavine, D. and Kanal, L., "Knowledge-Based Image Analysis", ETL-0258, USAETL, Fort Belvoir, VA, 1981.
22. STOCKMAN, G.C. and Kanal, L., "Problem Reduction Representation for the Linguistic Analysis of Waveforms", IEEE Trans. on PAMI 5, 287-298, May 1983.
23. SWAIN, P. and Davis, S., Remote Sensing - the Quantitative Approach, McGraw-Hill Inc., 1978.
24. WALTZ, D., "Understanding Line Drawings of Scenes with Shadows", The Psychology of Computer Vision, P.W. Winston (ed.), McGraw-Hill Book Co., New York, 1975.
25. WINKLER, R.L., "The Consensus of Subjective Probability Distribution", Management Sciences 15, B61-B75, 1968.
26. XIONG, F.L., Lambird, B. and Kanal, L., "An Experiment in the Recognition of Electrocardiograms Using a Structural Analysis Algorithm", Proc. IEEE 1983 Int'l. Conf. on Systems, Man, and Cybernetics, Bombay and New Delhi, India, December 1983.

THE AI RESEARCH ENVIRONMENT AT THE U.S. ARMY ENGINEER TOPOGRAPHIC LABORATORIES

Robert D. Leighty
Research Institute
US Army Engineer Topographic Laboratories

ABSTRACT

The U.S. Army Engineer Topographic Laboratories (USAETL) is responsible for Army research and development in the areas of mapping and terrain analysis. In general this involves methods, techniques, and systems for information processing related to the extraction, analysis, and presentation of terrain data. Typically, the data source is aerial imagery and the real-world information processing techniques for aerial imagery are very labor intensive. Previous research into automated techniques has not yielded results adequate to justify significant equipment developments necessary for future Army terrain information processing requirements. Thus, USAETL is making a significant commitment in AI with expectations for new and improved terrain information capabilities for the Army.

This paper presents
The following discussion will relate to the AI research environment at USAETL. This includes the rationale for the USAETL AI program, the objectives and approach of the Center for Artificial Intelligence (CAI), a description of the CAI AI facilities, and a brief description of the current CAI research program.

RATIONALE FOR USAETL AI PROGRAM

USAETL Mission

USAETL has, as a portion of its mission, responsibilities to accomplish research and development for the Field Army and the Defense Mapping Agency (DMA) in the areas of topographic mapping and terrain analysis. In general, the basic source of data for this work is aerial imagery. The basic problem associated with this data type is cost effective and timely extraction of information for the various tasks of the mapping and terrain analysis processes. Conventional approaches to

this problem area involve manual methods which are excessively labor intensive and time consuming. Future goals relate to automated systems necessary to address an increasing number of requirements for terrain data in military information and weapon systems.

Automated Pattern Recognition Research at USAETL

USAETL researchers have actively pursued automated extraction of information from aerial imagery with statistical pattern recognition techniques since the 1960's with only

limited progress in selected areas. These techniques have not been sufficiently robust or general to justify operational system development. Clearly, new approaches are needed for old problems as well as the new problems associated with digital terrain information processing for the Army of the future.

DARPA Image Understanding Program

In the mid-1970's USAETL began tracking the Defense Advanced Research Project Agency (DARPA) program in Image Understanding. In this program the Information Processing Techniques Office, DARPA, has contracted with artificial intelligence groups at universities such as MIT, Carnegie-Mellon University, Stanford University, University of Rochester, University of Maryland, Purdue University, University of Southern California, and SRI International (SRI), to investigate methods, techniques, and systems leading to useful automated machine vision capabilities. In the late 1970's DARPA began to focus its Image Understanding program on application areas and one application area was "cartography." The objective in this research area involved the extraction of information from aerial images for mapping purposes. For this effort, DARPA realigned a significant portion of its Image Understanding program to (1) attack fundamental problems in computer vision relevant to cartography and photo interpretation and (2) design and implement a testbed facility at SRI which would integrate software contributions from the Image Understanding contractors. In 1979, USAETL assumed the role as DARPA's agent for DARPA/DMA Cartographic Testbed.

USAETL Commitment to AI

USAETL began its commitment to AI in February 1981 with a decision to acquire a duplicate of the DARPA/DMA Cartographic Testbed for inhouse Research and Development programs. The rationale for this decision came from the need for improved techniques for information extraction from aerial imagery. It was reasoned that automated statistical pattern recognition approaches, with which USAETL researchers had considerable experience, would be needed in AI systems to generate symbols from image data. It was realized that the organization lacked AI programming expertise. However, through a proper mix of training, contracts, and new hires this capability could be developed over a period of time. Further, it was recognized that USAETL had experts in terrain analysis and automated cartography necessary for building knowledge-based systems. And, perhaps more significant, the cartographic and terrain data bases common to the mapping community could be employed in AI techniques to guide image information extraction processes.

CAI was established within the Research Institute in August 1982. The objectives of CAI are to conduct basic and applied research in artificial intelligence methods and techniques leading to semi-autonomous and autonomous systems of the future in support of USAETL mission areas.

USAETL Commitment to Army AI/Robotics

In March 1981, USAETL was requested by Deputy Chief of Staff, Research,

Development, and Acquisition (DCSRDA) to prepare a baseline Army Research and Development plan for AI/Robotics leading to techniques and systems to assist combat and combat support personnel in battlefield missions. Under a competitively awarded contract, SRI prepared and published a study report in May 1982 containing a suggested Army R&D plan. In July 1981, due to mounting interest in AI/Robotics within Army Headquarters, a DCSRDA Steering Committee for AI/Robotics was formed. This Committee contained representatives from Office of the Chief of Engineers (USAETL), Deputy Chief of Staff for Personnel (Army Research Institute for Behavioral and Social Sciences), DARCOM (Human Engineering Laboratory), The Surgeon General, and TRADOC and it opted to initiate an Army AI/Robotics Program. More than 100 applications of AI/Robotics systems to Army activities were prioritized by TRADOC Schools and Centers. Subsequent guidance from Army Headquarters was to concentrate on a small number of applications. This led to the Steering Committee selection of five "Demonstrators." USAETL prepared a plan for the Robotic Reconnaissance Vehicle with Terrain Analysis and this was combined with a Human Engineering Laboratory (HEL) plan. The objective of this demonstrator is to demonstrate, in two years after funding, the capability to plan and conduct teleoperated reconnaissance vehicle operations for representative battlefield missions. This plan was given top priority by TRADOC and an Army Science Board Ad Hoc Subgroup for AI/Robotics and has been funded for FY84 and FY85. USAETL will have responsibility for the hardware and software systems and HEL will be responsible for the demonstration scenario and the conduct of the demonstration.

THE CAI APPROACH

Objectives

To repeat, the objectives of CAI are to conduct basic and applied research in AI methods and techniques leading to semi-autonomous and autonomous systems of the future in support of USAETL mission areas. This implies that research will be conducted not only in information extraction from aerial imagery and autonomous vehicle systems within CAI, but also in support of the total USAETL Research and Development program, which includes Field Army, DMA, and Civil Works, as well as other USAETL customers.

Personnel

CAI currently has 13 professionals with technical backgrounds that include civil and electrical engineers, computer scientists, cartographers, geologists, foresters, and physicists. While all have advanced degrees, none have formal AI training above the master's degree; thus, retraining has a very high priority in CAI. The retraining will be available in several forms: long term, formal training at universities; part time, formal training at universities; AI short courses; TV lecture series; in-house contractor training; professional meetings; and self-learning on the in-house software/hardware systems.

CAI Facilities

USAETL will, by the end of FY83, have excellent AI facilities. The principal element of the Testbed hardware configuration is a DEC VAX-11/780 central processing unit. The VAX is a four-megabyte system with one tape

drive, two 300-megabyte disk drives, 16 teletype lines, floating point accelerator, and parallel DMA interface. The VAX interfaces directly to a variety of terminals, a digitizing table, a menu tablet, a Grinnell display system, a Versatec printer/plotter, and an Optronics color image scanner. A Symbolics Model 3600 LISP machine will be connected to the VAX system by an ETHERNET, as will other computer systems in CAI mentioned below. (Mention of commercially available equipment is not an endorsement of this equipment.)

CAI will have a number of software packages available for experimentation. To be of value, these must be tested and evaluated for potential use to the CAI program. The AI Testbed software from SRI will be thoroughly exercised so as to provide an effective interface between DMA and the DARPA Image Understanding Program. Other AI software packages, such as OPS5 and KES, will be available. By exercising the programs available with typical USAETL data, strengths and weaknesses of existing software will serve as the basis for the subsequent research program and acquisition of other software/hardware.

Approach

Most of the CAI effort for FY83 will be devoted to an on-going research program, training, testing of existing software packages, and tracking-related AI activities of other organizations. The on-going research program, which includes Computer-Assisted Photo Interpretation Research (CAPIR), AI/Robotics, etc., will be discussed subsequently. Training will consist of the formal and informal instruction indicated above.

Additionally, training exercises will be developed for small teams within CAI to serve as mechanisms to focus training activities. An example will be given subsequently. Efforts will be devoted to testing the DARPA/DMA Testbed software as well as other available software. Finally, related AI efforts of other governmental agencies will be tracked.

In FY84 CAI will begin to integrate AI into the on-going research program and develop a program which applies the DARPA/DMA Testbed capability to USAETL mission areas. The Army AI/Robotics Reconnaissance Vehicle Demonstrator will be funded and involve CAI in the demonstration and initiation of research leading to the autonomous vehicle.

CAI RESEARCH PROGRAM

Major elements of the FY83 on-going CAI research program will now be outlined. These efforts are in addition to other activities indicated above dealing with training and software testing.

Computer-Assisted Landform Analysis Program - CALAP

Operational terrain analysis from aerial imagery is labor intensive and requires expert terrain analysts. CAI has a research effort directed toward developing an interactive computer program that may be used to lead a relatively inexperienced photo interpreter through a landform analysis problem for any study area in a selected physiographic region.

CALAP conducts an interactive diagnostic dialogue with the interpreter and serves as a computer expert assistant. It operates analogous to a blind expert sitting with the interpreter. CALAP is based on the principles of physiography and geomorphology. The world is divisible into physiographic units and physiographic units can be divided into local areas based upon landform types. Any given physiographic unit contains only a small subset of the total landforms found in the world. Thus, if one is given the location of the aerial images to be studied, the physiographic unit containing this location is defined as well as an expected set of landforms to be found in that unit. The recognition diagnostics are then constrained to this expected set of landforms. In the analysis procedure the interpreter is prompted to look for and report terrain patterns. For example, if the study area is located in the Atlantic Coastal Plain he may be asked if a coastal shoreline exists in the study area. If so, this will indicate the possibility for coastal beaches, beach ridges and swales, sand dunes, etc., and after their characteristics are defined for the interpreter, the area adjacent to the shoreline is then searched for the expected set of landforms. The analysis would then move sequentially through decision trees associated with locating and delineating landforms expected to be found in association with tidal river basins, recent alluvium, and coastal plain terraces.

The CALAP program is presently written in FORTRAN and operates on a Hewlett Packard 1000/F-series minicomputer. The program has been recoded into OPS5 running in FRANZLISP on a DEC VAX 11/780 as a possible training aid in which FORTRAN and FRANZLISP code can be directly compared. While this is

instructive for the novice LISP programmer, the one-to-one recoding yields a less efficient LISP operating program.

Computer-Assisted Photo Interpretation Research - CAPIR

An in-house laboratory system has been designed, developed and used to support research studies and demonstrations of computer-assisted photo interpretation. This is called the CAPIR system. The focal point of the CAPIR system is a stereoscopic workstation incorporating an APPS-IV analytical plotter with graphic superposition, an integral voice recognition module, and two large application programs which support creation of geographic data bases directly from stereo images and manipulation and statistical analysis of the geographic data bases. The CAPIR system also incorporates a Data General Eclipse S-250 minicomputer with an Integral Array Processor and standard peripheral devices such as disks, magnetic tape drives, printers, and CRT display terminals.

CAPIR embodies three basic concepts: (1) direct data entry in a geographic coordinate system to digital files; (2) on-line stereodigitization using a computer-interfaced stereoscope; and (3) direct superposition of computer generated graphics in the stereomodel. Thus, points, lines, and areas with three-dimensional ground coordinates can be entered into the data bases and displayed in the working stereo images. Direct superposition provides a means to review, edit, and/or verify on-going or previously prepared digital data bases for the terrain areas covered by the stereo images. Solid-state CID cameras have been added to each optical channel of the stereoscope to enable subsequent computer

processing of the image under study by the photo interpreter. These capabilities, when interfaced to the AI Testbed, will provide the basis for an evolutionary image analysis capability leading to a system with a high degree of autonomy.

The CAPIR is presently an operational image analysis system with a manual capability for building terrain data bases. Integrating AI knowledge-based expert modules for location and delineation of landforms, drainage, vegetation, cultural patterns, etc., will provide growing capabilities for semi-automated image analysis. For example, an interpreter analyzing stereo images of an area might invoke an automated vegetation classifier by a voice command. This will cause the digital images to be sampled by the CID cameras and operated on by the vegetation expert software in the AI Testbed. The expert system would use existing information in the terrain or cartographic data bases to guide the digital area search and classification processes. Results are then displayed to the interpreter via the graphic superposition for his verification and/or edit. As more of the AI modules are added and as they get smarter (requiring less human intervention), the system will evolve to a semi-automated image analysis system. CAI is currently working on the next generation CAPIR involving a softcopy stereoscope.

Robotics

USAETL is interested in R&D leading to autonomous vehicle navigation in a battlefield environment. In the first stage of this effort, a vehicle will be teleoperated by non-line-of-sight communications from a control van. The vehicle will have a position/navigation inertial sensor, stereo cameras, and

communication equipment on-board. In the control van, super microcomputers will process and display the stereo images to the vehicle controller and plot the position of the remote vehicle as a blinking cursor on a graphics map background of another display. This map display will be used to plan and then operate the vehicle along a route selected by the computer and verified/edited by the operator. The route planner (expert system) will use digital terrain data bases to compute the best route between terminal and/or intermediate points for the route and the route is then displayed, along with the vehicle cursor, on the map display. Another display will be used for supplementary graphics that serve to provide the vehicle operator with additional information about the vehicle position in its surroundings. For example, this display could image digitally computed isometric images of the operational area with the vehicle plotted in proper position. It could show computer-generated images at points along the route where operator decisions are critical. Thus, turning points, change in slopes, and bridge and stream crossings are examples of critical points along the intended route that might have associated computer-generated images with which the operator can compare to the real-time stereo images and make steering adjustments if required. If the operator encounters difficulty not anticipated from the prior route planning, he can reenter the planning mode to navigate around the obstacle.

This is essentially the terrain navigation section of the Robotic Reconnaissance Vehicle Demonstrator, mentioned above, that is due to be demonstrated in late FY85. The long-term objective of the Demonstrator is to transform the human planning and system operating capabilities from the

control van to the vehicle. The vehicle would then have responsibility to plan its route from internal terrain data bases and navigate along this planned route with the aid of machine vision and local steering control. The vehicle will require smarts to recognize obstacles and plan alternate routes to the objective. The human supervisor would be needed only to assign missions, infrequently oversee operations, and be available to handle decisions and operations out of the range of the vehicle's potential.

Other Research Activities

There are two other smaller efforts requiring some mention. The first deals with a capability for automated delineation of drainage patterns from digital terrain elevation data and the second deals with building a prototype battlefield intelligence expert system that incorporates terrain data.

The drainage expert system is motivated from the need to make present labor-intensive manual cartographic processes of topographic drainage delineation for mapping purposes more efficient. Manual techniques require an operator to annotate topographic gullies under stereoscopic viewing conditions, often after terrain elevation data has been extracted for mapping purposes from the same area. Algorithms have been tested for application to drainage delineation from the digital elevation data. These will be incorporated into an expert system that will handle algorithm scheduling and control and integrate simple heuristics to operate in special case situations. This study will be done in the CAPIR environment with graphic superposition of delineated drainage directly into the stereomodel for operator verification/edit.

The battlefield intelligence expert system is a cooperative study between CAI and the U.S. Army Intelligence School and Center (USAICS) and is used by CAI to focus AI training and software testing activities. USAICS personnel will serve as the domain experts for the military intelligence and sensor management knowledge base building and CAI will serve as the AI experts to acquire and represent the knowledge in expert system form. The problem is associated with Intelligence Preparation of the Battlefield (IPB) and intelligence collection resource management. For an enemy area wherein we have knowledge of troop and equipment distribution and special areas of the terrain through which enemy units must pass to attack friendly positions, we are given some information about activity in one or more of the special areas. A hypothesis is then formed as to the nature of the activity and an optimum available sensor is selected and scheduled to acquire intelligence related to the hypothesis. The sequence of hypothesis and test leads to a conclusion of the enemy intent. This study has just begun and it is intended that available expert system building software such as OPS5, ROSIE, or EMYCIN will be used to test concepts.

SUMMARY

USAETL is making a commitment to AI research with the expectation that new and more efficient methods and techniques may be applied to the solution of old problems in its mission areas as well as the new problems associated with digital terrain information processing for the Army of the future. This commitment has involved allocation of technology based funds for equipping a state-of-

the-art AI research facility and establishment of a research group (CAI) with a charter to investigate AI for application within its mission areas. Additionally, a research budget and personnel have been assigned to AI research and start-up time is being provided for personnel training. Ongoing USAETL AI research efforts generally employ interactive (man-in-the-loop) approaches wherein AI modules are expected to provide efficient enhancements in an evolutionary manner rather than targeting upon specific AI-aided systems to be produced some years in the future.

This is the essence of the AI research environment at the U.S. Army Engineer Topographic Laboratories.

THE USE OF AI IN TARGET CLASSIFICATION

Dr. Durga Panda, Dr. Raj Aggarwal and Dr. Tod Levitt
Honeywell Systems and Research Center

ABSTRACT

A traditional approach to tactical target classification utilizes statistical pattern recognition techniques to classify targets segmented as single objects. Classification of tanks at four to ten kilometers in FLIR imagery is a typical example. There are two areas in which the traditional methodologies fall short. One is that these methods do not readily generalize in order to recognize complexes of many structures, such as missile sites and power plants. The second is that, even within the domain of single blob target classification, it is difficult to incorporate contextual information, for example, the fact that tanks do not appear in the sky, into the statistical pattern recognition approach.

Obviously, an ideal target classification system can account both for recognition of complex objects and system methodologies that readily utilize contextual information. Advanced applications for these capabilities include high-value target recognition, passive terminal homing systems, sentry robots, bomb damage analysis, intelligent remote surveillance, landmark based guidance, and autonomous (and semi-autonomous) tactical vehicles.

In this presentation we focus on the first area of generalized (multi-object) target classification. Artificial Intelligence (AI) methods provide an approach to target classification that uses knowledge representation and manipulation techniques to extend the base given by statistical pattern recognition.

INTRODUCTION

Traditional approach to target recognition consists of three simple processing steps. First, the image is segmented, then, in the second step, features are extracted from the segmented objects, and lastly, based on statistical models, the features are classified into various classes. Figure 1 shows the three steps. The segmentor, the feature analyzer, and the classifier are designed based on statistical analysis of the training

data. This provides classification of a single isolated object at a time. The single target classifiers are statistical classifiers such as Bayes, K-nearest neighbors, linear discriminant, and so on (1).

These systems do not take advantage of any embedded information in the scene context, other than the individual segmented object in a given frame. These systems do not have provisions for making use of any object-to-object interrelationship type of information. Information present in

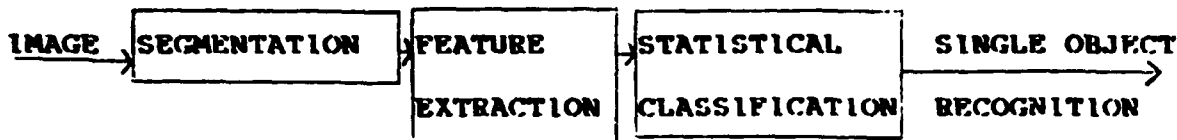


Figure 1. Conventional Target Recognition Consists Of Three Major Steps

the surround of the segmented object which could confirm or refute the presence of a target is not used in the conventional approach to target recognition. Consistency of the inferences made in individual image frames in time sequence is typically not used in the single frame target recognition approach. The time sequence analysis can provide object motion information for target recognition which cannot be obtained from single frame processing. Also, when the test data somewhat deviates in characteristics from the training data used for the analysis and design of the classifiers there is drastic performance degradation.

Artificial Intelligence (AI) techniques can be used to improve the target recognizer by filling some of the gaps mentioned above. AI can provide this assistance in three categories:

- spatial context analysis
- temporal context analysis
- knowledge based executive control.

The following sections describe these three categories and their impacts on the target recognizer.

SPATIAL CONTEXT ANALYSIS

Target recognition can be aided by context that helps resolve conflicts. Examples of such conflict resolution rules are the facts that the tanks do not appear in lakes/ponds or sky. The sky and water signature being different from open terrain signature, it can be distinguished by a low level vision subsystem.

Spatial context also enables us to recognize target formation such as convoys (2). Identification of target components with the use of spatial context information enables appropriate aim point selection. Other usage of spatial context is in general terrain interpretation. For example, the presence of a road is a contextual clue that helps localize the search for targets such as 2nd echelon supply or reinforcement vehicles in deep strike and battlefield interdiction missions. Other examples of such contextual cues are road intersection and road and river intersection (location of strategic bridge).

Representation of knowledge is one of the most important requirements of spatial context information utilization. The knowledge base models the world, models the environment, and

at the lowest level of hierarchy, models the components of a target of interest. The knowledge can be relational information, rules and constraints, a specific instantiation of a target of interest, typical attributes of a target or a scene component, and finally chances (probabilities) of occurrence of certain scene contents given the evidence of certain attributes. This last piece of knowledge, the chance of occurrence, is important in making use of the knowledge base that is modelled in the system. The occurrence information is used in deducing inference, most often from incomplete information, about the scene context.

The representation is best done in a declarative form. The hierarchical model of the world is declaratively represented in terms of sets of nodes, each of the nodes representing a component of the scene or of the target of interest. The nodes that represent interrelated components are linked by arcs, thus forming a hierarchical graph representation of the world (3). Figure 2 shows an example of an AND/OR graph.

In Figure 2 a truck is modelled by a graph of n alternative representations, each of the alternatives being a graph (called a subgraph in the figure) in itself. The n -th representation, for example, shows that the truck may consist of an engine and a body, with a specific global (i.e. inter-object) relationship, denoted as "Global n " in Figure 2. The models of the engine and the body, in turn, are further hierarchical subgraphs. Thus, this representation allows modelling of the world of the environment in terms of its relevant constituents, each of the constituents being modelled by their subconstituents, and so on, down to the level of individual target components. This allows recognition

of complex multi-object targets such as bridges and powerplants (high value targets) as well as recognition of target components such as tank tread covers. This target component recognition provides a more effective critical aimpoint selection and higher probability of kill than is possible by an aimpoint conventionally selected from the centroid of the object (4).

TEMPORAL CONTEXT

Contextual information is also available in another dimension, which is time. Time sequence histories of detected and recognized targets have been valuable in reducing false alarms. General pixel level time sequence disparity analysis provides optical flow information in the scene (5). Information contained in optical flow enables sensor transformation, and as a result, tracking, hand off, etc., are done more efficiently. Lastly, temporal context also enables the detection of moving objects, which is a very valuable cue in target recognition.

In principle, temporal context analysis provides symbolic accrual of information over time. An important part of this is the disparity analysis leading to frame-to-frame correspondence of scene components. Symbolic matching is a computationally inexpensive way of analyzing the disparity. It computes disparity between object pairs (rather than pixel pairs) between two time sequence frames. For each candidate object a local area search is made. The transformation that gives the best matching object in the search area is mapped to Hough domain. Clusters in the Hough transform give the overall translation, rotation, and scale change parameters between the frame pairs.

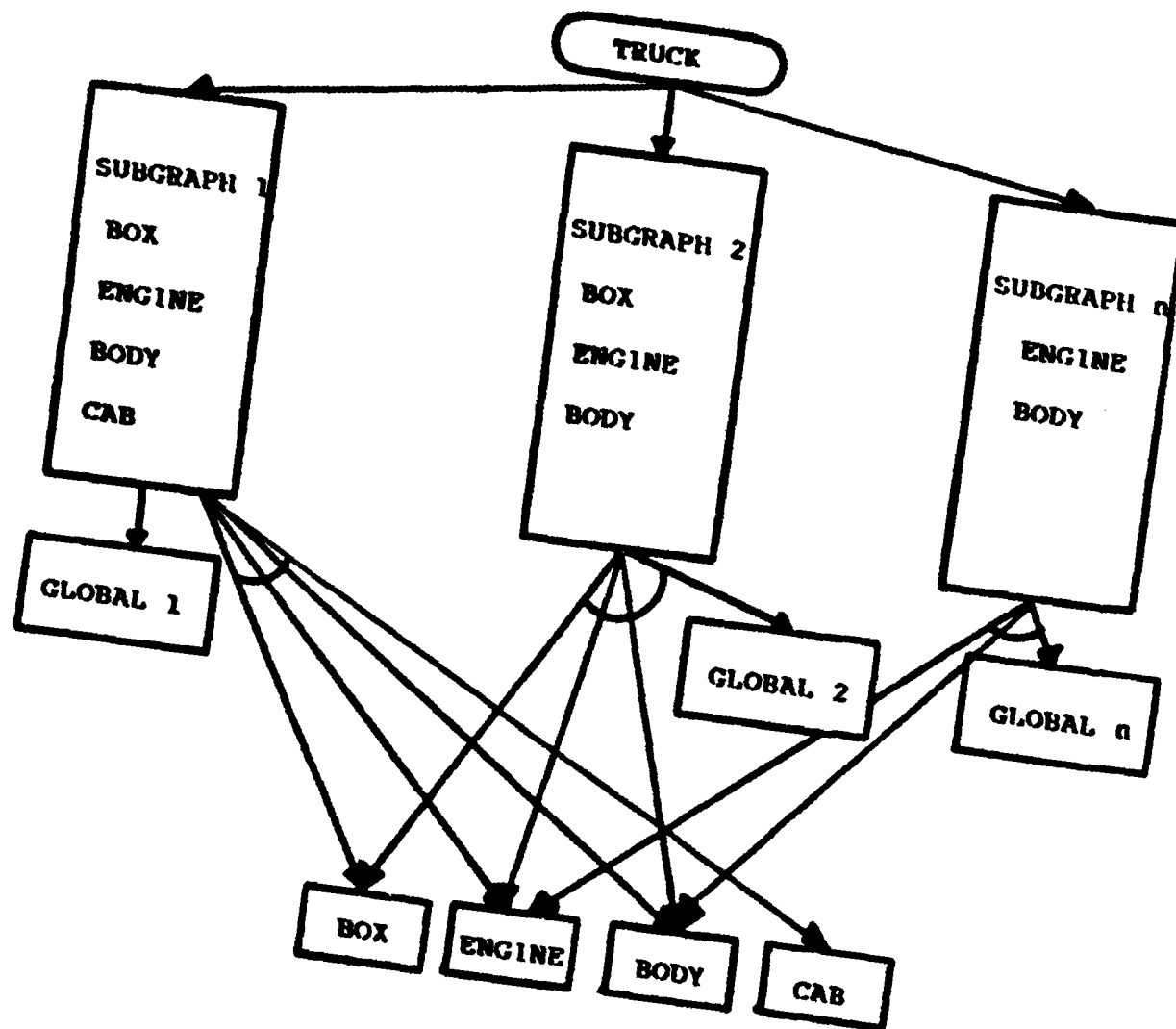


Figure 2. Knowledge Representation by AND/OR Graph

After the objects are matched the semantic information about the objects in the successive time frames is accrued. The semantic information includes object characteristics and single frame inferences related to the objects. This accrued evidence confirms object classification results and resolves any conflicts in frame-to-frame classification via heuristic programming. The location information about the matching object pairs enables detection of moving objects, as these are the objects whose frame-to-frame transformation differs significantly from that of the entire frame.

KNOWLEDGE BASED EXECUTIVE CONTROLLER

Dynamic selection of algorithms and algorithm parameters is a useful way of optimizing the target recognizer to the particular scene being analyzed. Knowledge based control is one approach to this dynamic system optimization. In this approach, all the algorithmic modules are controlled and manipulated by a knowledge based controller. The controller analyzes the incoming signal and the intermediate results (Figure 3) for algorithm and parameter selection.

The knowledge base is a set of selection rules. These selection rules are based on expertise gained during the training and development of the target recognizer algorithms. The selection rules can be set in a production rule mode, where the outputs or the actions of the rule set are:

- selection of appropriate window size.
- selection of appropriate operator type.

- selection of appropriate values for the cut off parameters (i.e., threshold, etc.).

The conditions that trigger these actions are typically measurements made on the input image itself or on the intermediate results of the target recognizer. Simple examples of the measurements on the input image are signal-to-noise ratio, local contrast, and global contrast. Examples of intermediate results are number of target size clutters that are segmented by the segmentor and frame-to-frame inconsistency of the segmented objects.

The knowledge based executive controller provides an effective means of maintaining the target recognizer performance from one scene type to another through dynamic optimization. In cases where scene conditions are extremely unfavorable the controller provides graceful degradation of performance and saves the recognizer from being practically inoperative.

CONCLUSION

Conventional target recognizers have limited functional and performance capability. The use of AI in the areas of algorithm control, spatial context, and temporal context increases the usefulness and functional capability of a target recognizer. Spatial context processing in addition to single object classification gives additional clues about the scene content and improves target detection/classification. Temporal context provides optical flow and object motion information, and enables accrual of target recognition results over time to help resolve conflicts. Knowledge based executive control maintains target recognition performance even when test data significantly differs from the training data.

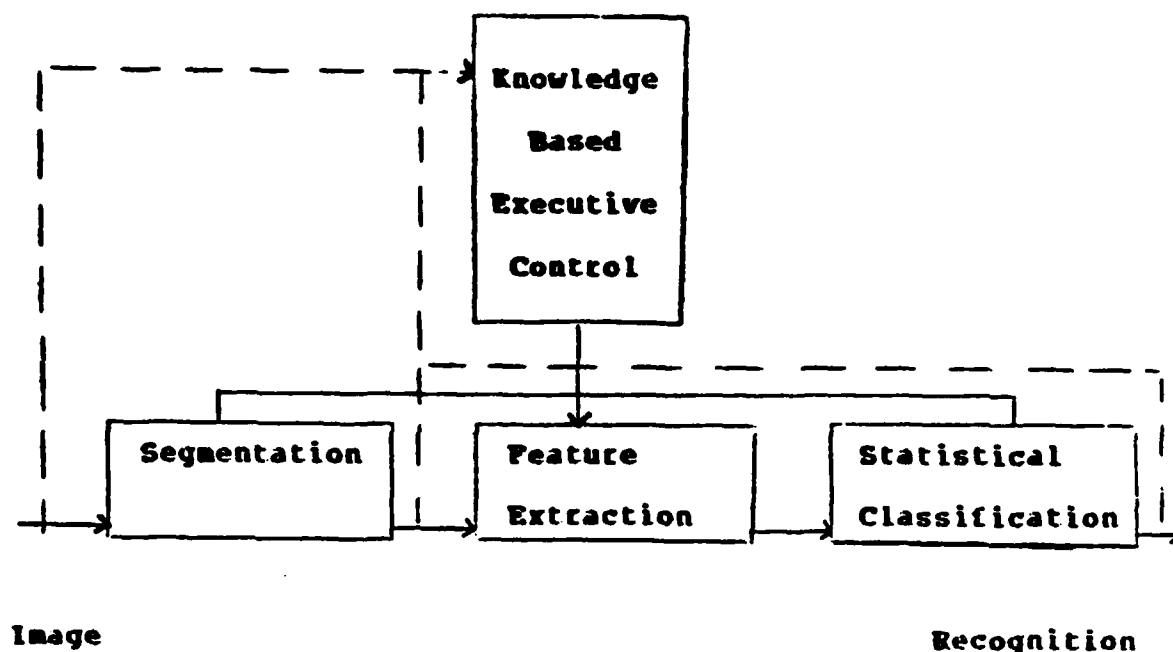


Figure 3. Knowledge Based Executive Control of the Algorithmic Modules in Figure 1

REFERENCES

1. Infrared Technology for Target Detection and Classification, Proceedings of the SPIE, Vol. 302, August 1981. (This provides a good overview of the state of the art of various conventional target recognizers being developed.)
2. Durga P. Panda, et al., Automatic Image Recognition System, Final Report of the DARPA/AFWAL contract on Image Understanding, no. 79SRC86, February 1980.
3. N.J. Nilsson, Principles of Artificial Intelligence, Tioga Press, Palo Alto, 1980.
4. R.K. Aggarwal and T. Wittenburg. Syntactic Recognition of Tactical Targets, Proceedings of Image Understanding Workshop, November 1978.
5. S.T. Barnard and W.B. Thompson, Disparity Analysis of Images, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, November 1980.

AI CONTEXT ANALYSIS FOR AUTOMATIC TARGET RECOGNITION

Andrew J. Spiessbach and John F. Gilmore
Engineering Experiment Station
Georgia Institute of Technology

ABSTRACT

Research directed toward the goal of integrating contextual scene information into the automatic target classification process is described. The basic approach adopted is an information fusion and feedback architecture centered around an artificial intelligence (AI) production system. A parallel organization of specialized algorithms extracts complementary contextual discriminants from the local, global, and temporal image data. A high-level relation data structure integrates this imagery information with non-imagery scene data, such as a priori knowledge of terrain, environment, and expected threat, thereby providing a symbolic representation of the total dynamic scene. Extensive domain-specific and general knowledge applied to the scene representation by an AI Expert System exploits the associative evidence to deduce additional facts, infer signatures, and perform collective target class decisions. In addition, an AI global/local control strategy routes derived knowledge and revised hypotheses to all extraction algorithms to bootstrap overall preprocessor intelligence and provide adaptive optimization through feedback.

INTRODUCTION

The complexity of the modern battlefield dictates the need for automating the image understanding process. As more sophisticated weapons, vehicles, and tactics are developed to cope with this high-threat environment, mission effectiveness inevitably becomes limited by the information bottleneck at the sensor/human interface. A high level of machine intelligence is needed to assist human operators in targeting, navigation, and situation assessment functions, and to replace the human under high vulnerability or weight/volume constrained conditions.

Machine decision aids for automatic target classification are now being vigorously pursued by the Army to improve operational performance and enhance survivability. Several such systems have (1,2) already been developed using image processing and statistical pattern recognition technology. While existing algorithms for automatic target recognition (ATR) have demonstrated adequate performance for restricted test situations, this performance is unfortunately not extensible to larger, realistic scene domains.

Context provides the essential flexibility needed to extend the

domain in which target recognizers work from the controlled environment of the laboratory to the real world of scene variability. Whereas moderate gains in automatic target recognition performance have been achieved in the past by exploiting only static signatures localized near the target, context-based classification promises a new generation of machine intelligence by capitalizing on all synergistic relationships available in the total dynamic scene.

Exploitation of contextual information provides both the opportunity and the incentive to simultaneously improve target classification performance and reduce processing requirements. The additional target knowledge brought to bear on the decision process results in enhanced classification accuracy and ensures robust performance with scene variability. Scene history, available in image temporal changes as well as a priori scene expectations, make it possible to adaptively tune all algorithms for the scene at hand. Knowing what to expect reduces the recognition problem and, consequently, the processing burden. In addition, the availability of additional information sources can be used to bypass intensive computational steps and permits the use of more efficient data structures and operators.

THE GENERALIZATION GAP

The current generation of ATR technology has demonstrated high levels of performance for limited sample sets and restricted problem domains. Unfortunately, this performance is not extensible to the highly variable scene conditions of a realistic battlefield. Existing algorithms are extremely data dependent and inflexible. This is

a fundamental consequence of conventional object-based pattern classification, where the basic paradigm is to create a matched-filter to the measurement statistics of a database of prototypical samples. In practice, the enormous combinatoric possibilities of real world scenes prohibits generating and training with a database that is statistically representative of a general tactical scenario. Even if algorithm training with sample images was feasible, it would not, in general, be sufficient. Object-oriented statistical classification requires more precise and complete data for feature extraction than is typically available from a tactical image.

The major challenge of ATR development is then not so much improving performance but rather maintaining performance while simultaneously expanding the domain of applicability. Therefore, the pivotal ATR technical issue is algorithm robustness, that is, finding approaches that work not only for specific instances but also for the general case. Bridging the generalization gap between the restricted domain of the laboratory and the real world of scene variability is the final obstacle to widespread ATR deployment.

The good news is that there is only one unresolved problem in target recognizer technology, i.e. generalization. The bad news is that its solution requires not just new algorithms, but an entirely new approach.

AI FOR TARGET RECOGNIZERS

The ultimate goal of autonomous acquisition algorithm research is to achieve, with a machine, a target recognition capability equal or superior to that of a human.

Achieving this goal demands that machines exploit the same contextual information that human visual perception absolutely requires. A high-level scene analysis capability, using artificial intelligence, is fundamental to context exploitation.

A context-based approach to automatic target recognition can be characterized by three conceptual stages of processing that correspond to a hierarchy of increasingly higher levels of scene description:

1. Image processing - Signal-level operations on individual pixels
2. Pattern recognition - Attribute measurement and classification of individual, large-scale scene components
3. Artificial intelligence - Symbol-level processing and analysis of the total dynamic scene.

These three categories are analogous to machine language, assembly language, and higher order languages in computer systems. High level scene descriptions provide the target recognizer with improved efficiency and flexibility in the same way that higher order languages do for computers.

By providing the pivotal mechanism to relate disparate scene elements to each other and to world models as well, AI permits the exploitation of additional data and knowledge sources for accurate target classification. Since the final decision processes are performed in the abstract space of symbols rather than at the combinatorially explosive signal level, performance sensitivity to scene variability is dramatically reduced. High-level processing also provides the transparency needed for flexible control. In addition, symbolic representa-

tions of sensed data are directly compatible with the way humans express and understand world concepts, thereby greatly enhancing the types of knowledge that can be effectively utilized in the system. For example, scene attributes rather than image attributes can be used for discrimination, so that consequent decisions are based on physical properties and not mathematical artifacts.

ATR RESEARCH REQUIREMENTS

Image understanding requires both adequate information about the scene and the know-how to draw correct conclusions from it. As shown in Figure 1, the overall goal of robust decision making can be represented as the conjunction of these two subgoals. The quantity and quality of available scene information *must provide sufficiently discriminating evidence to support the correct hypotheses while simultaneously refuting implausible or irrelevant alternatives.* Domain-specific and general problem-solving knowledge must provide the judgment capability needed to resolve conflicts and converge on the truth.

There are two prerequisites for satisfying the adequate information condition. First, sufficient raw data about the scene must be collected and input into target recognizers. Data quantity is a pivotal issue which has presented conventional ATR designers with a serious dilemma: not enough data to achieve performance; too much data to be practically processed. Secondly, the ATR system must have the "intelligence" to reliably distill the necessary and relevant high-level information from the raw sensor data. This poses a second dilemma: too much data and not enough information.

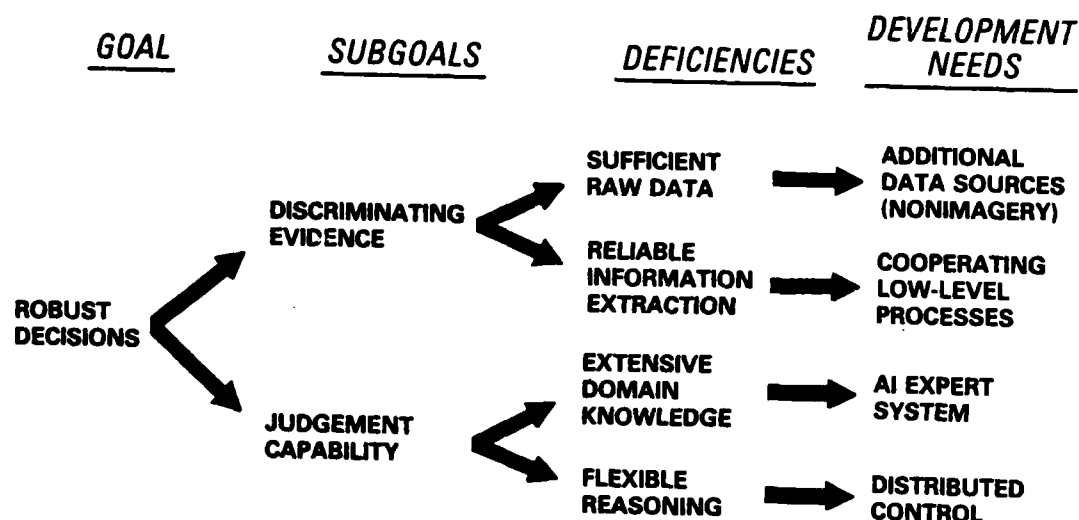


Figure 1. Research Goals for Advanced Target Recognizers

Additional sources of raw data are available to a target recognizer in a tactical battlefield (Table 1) and should be exploited. Useful nonimagery data such as a priori knowledge of terrain, environment, and expected threat must be input into target recognizers. Imagery data that is input but currently ignored (temporal, global, structural) must be fully exploited. The requirement for processing so much additional data poses a significant challenge to pragmatic hardware implementation; however, all perceived difficulties are more than offset by the abundance of constraints introduced by scene context to solve the problem. The introduction of independent, complementary scene data provides all additional relationships necessary to resolve any uncertainties or conflicts. Exploiting constraints contained in scene attribute relationships not only provides accurate decisions but also makes it possible to use efficient

model-driven approaches in place of data-dependent methods.

Properly exploited dynamic scene data can, in fact, reduce the processing burden even though more data is initially input. The inherent potential exists to accumulate scene history for scene prediction and feedback, as well as to exploit a priori data as cues to adaptively optimize algorithms for the instantaneous scene at hand. The use of additional information to reduce the problem that the algorithms must solve provides both the opportunity and incentive to simultaneously improve both target classification performance and implementation feasibility. It is the synergistic quality of the additional scene discriminants that reduces the quantity of processing required. In effect, the additional knowledge of the past evaluations and scene expectations boosts the intelligence level of the preprocessor and enables

it to reject nonessential raw data and to precisely isolate all necessary target data.

The introduction of feedback control in target recognizers also solves the problem of extracting the right information from the raw data. Hypotheses and additional facts generated by a high-level AI scene analysis of all information using extensive domain knowledge can be fed back to the early stages of processing. Therefore, by effectively bringing the system's intelligence closer to the sensor, only the most relevant and necessary information is

extracted from the mass of raw data. In addition, to facilitate the conversion of signal data to symbolic discriminants, specialized algorithms, each separately optimized for complementary functions, are needed to decompose the scene into component elements and isolate the temporal, global, and localized discriminant information using a parallel processing architecture. Figure 2 illustrates this concept of distributed problem solving (3) where data is abstracted into intrinsic scene constituents which can share information to effect cooperative low-level processing.

• LOCAL	• OBJECT BOUNDARIES (SIZE, SHAPE) • INTERNAL/EXTERNAL STATISTICS (CONTRAST, BRIGHTNESS, INTENSITY DISTRIBUTION) • INTERNAL STRUCTURE (TEXTURE, TOPOLOGY, HOT/COLD REGIONS, STRUCTURAL PRIMITIVES)
• GLOBAL	• NATURAL REGIONS (SKY, WATER, FOREST, FIELD, OBSTRUCTIONS) • CULTURAL REGIONS (ROADS, BUILDINGS, BRIDGES, AIR FIELDS) • LINEAL PATTERNS (HORIZON, FENCES, TREE LINES, VEHICLE TRACKS) • OBJECT PATTERNS (CONVOYS, TARGET ARRAYS, MINEFIELDS)
• SCENE DYNAMICS	• PLATFORM MOTION (MOTION STEREO FOR RANGE/3D RELIEF) • OBJECT MOTION (MTI, VELOCITY, ASPECT) • TEMPORAL STATISTICS (SEQUENTIAL COMPOUND DECISIONS, NOISE SUPPRESSION) • SCENE HISTORY (SIGNATURE PREDICTION, ADAPTIVE ALGORITHM OPTIMIZATION)
• ANCILLARY DATA	• TERRAIN (DMA DATA BASE, RECCE REPORTS) • ENVIRONMENTAL (WEATHER, SEASON, TEMPERATURE) • INTELLIGENCE (TACTICS, DOCTRINE, COUNTERMEASURES, THREATS) • SCENARIO CONSTRAINTS (MISSION PROFILE, TIME OF DAY, SYSTEM CHARACTERISTICS) • OTHER SENSORS

Table 1. Additional Sources of Scene Data

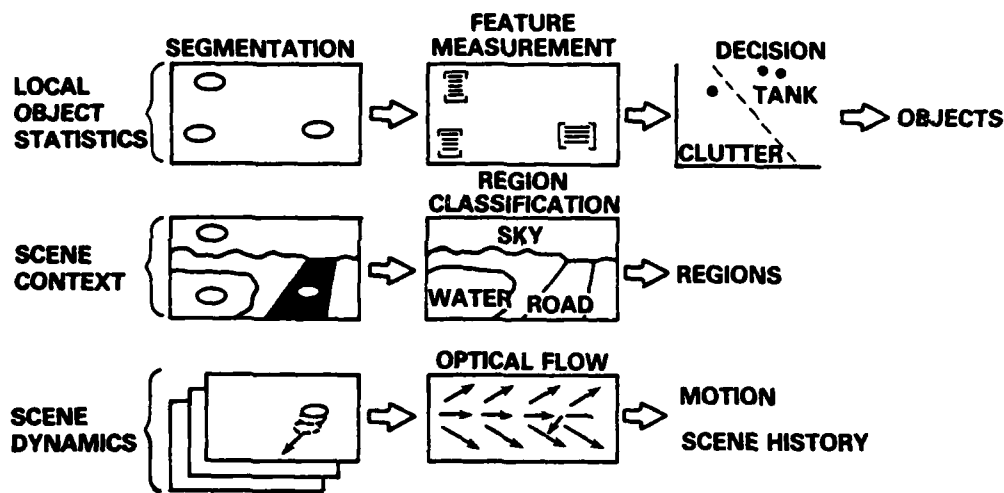


Figure 2. Cooperative low-level processing for intrinsic information extraction

While individual algorithms are important, the key to context-based target classification is the fusion of all information. By its very nature, context is an integration process, a coherent relation of complementary data. The real payoff for target recognition is the synergistic exploitation of disparate information for a single collective decision. AI fusion derives its accuracy from the consistency of independent, corroborating data and from the power of association. When making judgments based on evidential information, the whole is greater than the sum of its parts.

Figure 3 illustrates the combined information fusion and feedback process. Component information extracted in the parallel algorithm paths is integrated and related in the artificial intelligence scene description and synergistically exploited for collective target classification. The artificial intelligence system also serves as a clearing house for derived and substantiated knowledge used to

optimize the extraction algorithms. The fusion output serves, in this way, to improve system performance while simultaneously minimizing processing.

The extraction, fusion and feedback of information requires knowledge. Extensive and explicit knowledge at the level of human experts is required in the domain of tactical missions and scene analysis. This necessitates the incorporation of AI Expert System technology into target recognizer systems. However, base-level knowledge alone is insufficient. The diverse types of data and specialized algorithms inherent in generalized computer vision systems most naturally fit within a heterarchical system organization. Distributed control networks are most suitable for implementing such systems of cooperating, specialized subsystems, and require a meta-level architecture (4,5) with expert knowledge in the domain of information processing. Such meta-knowledge takes the form of introspection: knowledge about the system's algo-

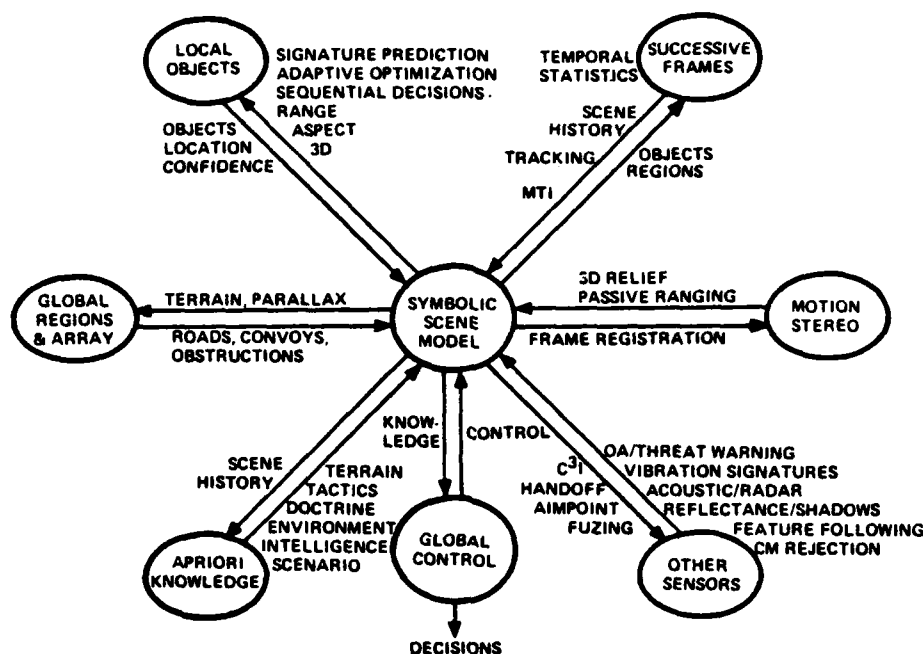


Figure 3. Information Fusion and Feedback Concept

rithms (capabilities, limitations, requirements); planning knowledge to alter processing sequences and information flow; knowledge about base-level knowledge. Therefore, flexible image understanding requires domain-specific knowledge characteristic of the ATR design expert as well as that of the copilot/gunner or photointerpreter.

SYSTEM CONCEPT

An information fusion and feedback concept is being pursued which is centered around a rule-based production system. A parallel architecture of image processing and pattern recognition algorithms converts signal data into symbolic information, using specialized techniques for each complementary scene attribute. An AI

Expert System fuses the extracted information into a total scene description and applies rule-based inference to perform target classification. An AI global/local control strategy provides the benefits of synergistic context exploitation to all algorithm elements by the feedback of derived, highly confident knowledge to adaptively optimize algorithm parameters. Figure 4 shows how this approach exploits all information in the dynamic scene, non-imagery as well as imagery data, for a collective target classification decision, and then feeds back hypotheses to sequentially refine the accuracy of the extraction algorithms.

The challenge of rapid, coherent manipulation of such large quantities of scene data is met by a symbolic representation of all information using an AI high-level data structure. A highly ordered search and decision

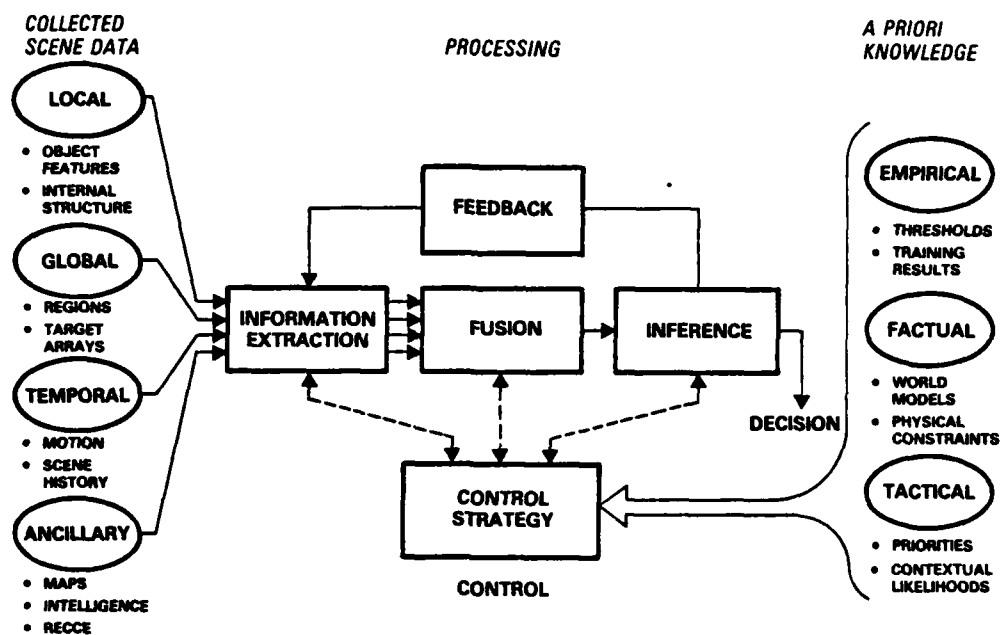


Figure 4. Advanced Target Recognition System Concept

strategy provides accurate, rule-based target classification, using an extensive knowledge of explicit physical constraints, tactics, doctrine, and common sense reasoning concepts dramatically augments the implicit empirical knowledge numerically contained in discriminant function coefficients and algorithm thresholds in conventional statistical approaches.

Figure 4 explicitly demonstrates the dual roles of scene data and knowledge base in the target classification process. The knowledge base provides the standard for comparison and evaluation of scene data as well as the basis for converting support into degrees of belief in the various alternative class decision possibilities. It is essential that scene data and knowledge base be treated with equal emphasis in context-based target classification. This consideration has

been given singular importance in formulating this system concept. All system extraction algorithms are specifically geared to transforming scene data and knowledge into a coherent, common framework for analysis and decision.

FUNCTIONAL DESCRIPTION

The baseline system approach consists of a modular algorithm architecture centered around an AI production system. The rule-based production system was selected over alternative pattern-directed inference approaches because of its superior ability to use the extensive knowledge required for accurate context exploitation. The central data structure of the production system contains all information in the scene in a high

The AI Expert system reasoning approach is shown in Figure 6. After the high-level scene description is constructed, the representation is checked using general knowledge of scene and image relationships to identify inconsistencies and resolve conflicts. After the constraint satisfaction (6) process has validated the scene information at a coarse-gram level, this information is used as axioms to derive additional evidence. Forward chaining of production rules with AND/OR antecedents is used for deductive inference and fact finding. Each initial or derived fact is associated with a belief function (7),



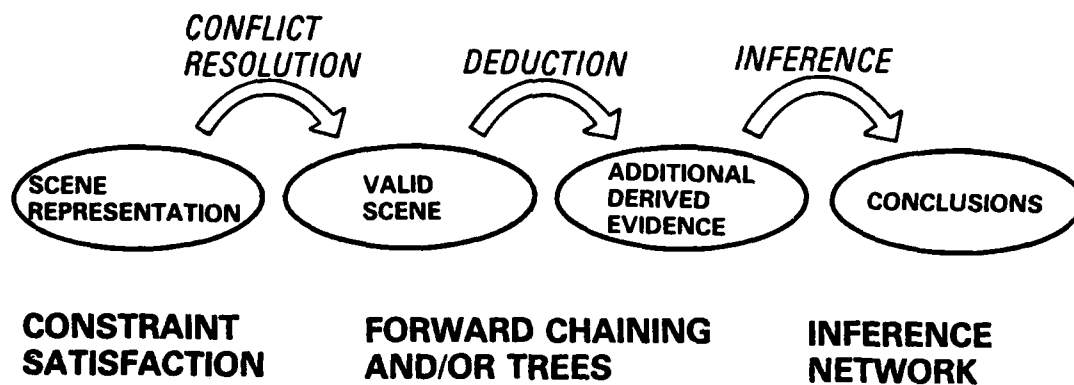


Figure 6. AI Expert System Inference Mechanisms

that maps each of the classification possibilities and their relevant combinations into a numerical degree of partial belief. All the evidence generated by the rewrite rules is pooled using Dempster's rule of combination (8) to draw final conclusions on target classifications.

Total synergistic exploitation of scene context is achieved by a feedback approach using a system of global and local control. As shown in Figure 7, the global control strategy provides overall system task management and adaptively determines optimal processing sequences. It also serves as a channel for the distribution of ancillary input data about the expected scene, and high confidence hypotheses and confirmed knowledge generated in the AI context analyzer. In this way, all information in the scene is made available to each of the individual context exploitation processes in the system.

Six separate, local control strategies are linked in a coherent network by the global system controller:

1. Local statistical control - performs algorithm selection and

adaptive optimization for the local object statistical algorithm chain (prescreening, segmentation, feature extraction, and classification).

2. Syntax control - selects one of three available structural classification approaches on the basis of range data extracted by motion stereo.
3. Global control - adaptively optimizes region segmentation and classification algorithms.
4. Temporal control - provides the decision strategy for velocity extraction, sequential compound decisions, parameter estimation, and motion stereo algorithms.
5. Clustering context control - directs the analysis of target arrays.
6. Context analyzer control - provides strategies for building and analyzing the scene representation, collective target classification, and feedback hypotheses.

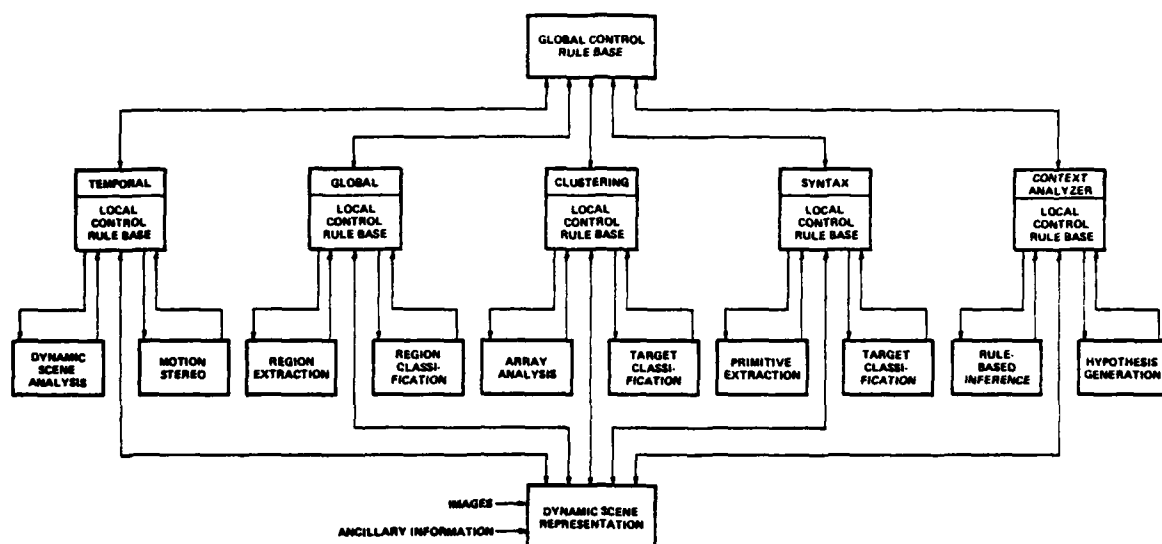


Figure 7. Distributed problem solving network of rule-based controllers

SUMMARY

The concept and goals of a context exploitation system for automatic target recognition was described. A specific system implementation responsive to these requirements is concurrently under development. The flexibility of this system provided by AI technology promises robust performance for a wide range of image understanding applications.

ACKNOWLEDGEMENTS

The work reported herein was supported in part by the U.S. Army Night Vision and Electro-Optics Laboratory under contract DAAK70-82-C-0215. Research was conducted at Martin Marietta Aerospace, Orlando, Florida where both authors were formerly affiliated.

REFERENCES

1. Soland, D. and P. Narendra, "Prototype Automatic Target Screener", Proceedings of SPIE, Vol. 178, pp. 175-184, 1979
2. Helland, A.R. and G.E. Tisdale, "The AUTO-Q Digital Image Processor for Autoprocessing of Reconnaissance Images", NAE-CON '80 Proceedings, p. 102, 1980
3. Davis, Randall and Reig G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving", Artificial Intelligence, Vol. 20, pp. 63-109, 1983
4. Genesereth, Michael R., "An Overview of Meta-Level Architecture", National Conference of Artificial Intelligence, Washington, D.C., 1983
5. Hayes-Roth, Frederick, Donald A. Waterman and Douglas B. Lenat, Building Expert Systems, Addison-Wesley Publishing Co., Reading, Massachusetts, 1983
6. Rich, Elaine, Artificial Intelligence, McGraw-Hill, New York, New York, 1983
7. Shafer, Glenn, A Mathematical Theory of Evidence, Princeton University Press, Princeton, New Jersey, 1976
8. Dempster, Arthur P., "A Generalization of Bayesian Inference", Journal of the Royal Statistical Society, Series B, Vol. 30, pp. 205-247, 1968

INTEGRATED DSS DEVELOPMENT TOOLS FOR MICRO COMPUTERS

DR. CLYDE W. HOLSAPPLE AND DR. ANDREW B. WHINSTON
Management Information Research Center
Purdue University

The opinions expressed in this manuscript are those of the authors and do not necessarily represent those of Purdue University.

INTRODUCTION

With the rapid developments in the computer field during the last 30 years we are clearly in the midst of what has been referred to as the information age. Both the volume and complexity of information processing to support individual and organizational decision making has rapidly increased. Clearly the technological innovations in the computer field will create the potential for even greater emphasis on computer based support of decision making. However, the key challenge for the future, as it has been in the past, is the provision of software so that the integration of human thinking and computer processing can be done in an effective manner. Efforts to understand how best to configure hardware and design generalized software which can facilitate the rapid and cost effective construction of computer based systems to aid in decision making has lead to the development of a new field - Decision Support Systems (DSS). While sharing the goals of the earlier Management Information Systems and Data Base Management, namely the efficient storage and selective retrieval of data, DSS attempts to go further with the integration of algorithms that can generate data as needed and a greater

emphasis on integration of human thinking through menu selection and forms capability. DSS places emphasis on helping the decision maker in dealing with less structured environments than MIS area had attempted.

FRAMEWORK FOR MICRO BASED DSS

Before considering software tools for building a Decision Support System (DSS) it is important to develop a general understanding of the generic structure of a DSS (1). The approach we shall use contains three components: a language system (LS), a knowledge system (KS), and a problem processing system (PPS). The three interrelated components and their relationship with a decision maker are illustrated in Figure 1.

We shall refer to the LS as the collection of all facilities that permit the decision maker to interact with the computer system. Such facilities include linguistic capabilities to express data requirements and the ability to express statistical analysis as examples. Besides a linguistic interaction a more pictorial capability would consist of forms management including the well-known spreadsheet form.

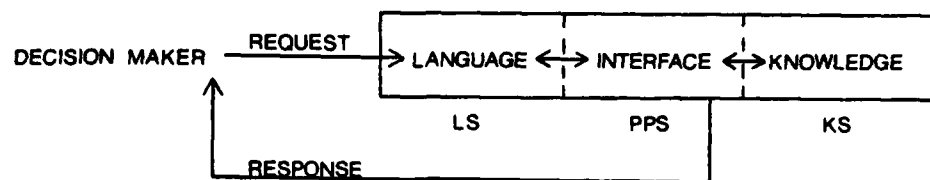


Fig. 1 Structure of a Decision Support System

In order to have an effective DSS in a particular area of application, knowledge specific to that application must be managed by the computer system. The organization of the knowledge is important for the facile operation of the LS. As we shall see, the potential types of knowledge include empirical facts, derived knowledge based on formulas, and procedural knowledge which embodies an algorithm for computing facts.

The final component is a problem processing system. A main function of a decision support system is to recognize and process sentences or expressions of the LS and then to find and produce the required information found in the KS. The PPS must be able to handle the variety of manners of expression in the LS as well as the different kinds of knowledge contained in the KS.

DSS, by its very nature, implies one person or a small group with a common goal interacting with a computer system dedicated to facilitating goal achievement. Thus it is natural that with the recent introduction and rapid growth of the new generation of 16 bit micros is an associated growth in the interest in building DSS based computer systems. Software tools to build micro based DSS systems are now being introduced.

In this paper we shall review such tools in light of our conceptual framework for DSS.

STRATEGY FOR BUILDING MICRO BASED DSS

Because of the micro computer revolution, computers are becoming indispensable tools for many, if not most, decision makers. While micros have tremendous potential, their actual usefulness to a decision maker is largely dependent on the nature of available software. In certain cases appropriate application software is available for supporting all or most of a decision maker's needs. In cases where such software is unavailable, a DSS developer can select a generic tool specifically intended for a particular kind of task and prepare that tool for the needs of the decision maker. Alternatively, the tool may not require the expertise of a professional developer, so that a decision maker can use it directly. For information storage and retrieval, there are file handlers such as dBase and Condor. For spreadsheet analysis there are packages such as VisiCalc. Other types of modeling needs would warrant a programming language such as PASCAL.

While these kinds of tools are of value, they represent piecemeal approaches

to building a DSS. This recognition has led to the recent emergence of a new generation of tools which integrate a variety of information processing tasks into a single tool for building a DSS. Such systems can greatly facilitate a DSS builder's ability to bring about knowledge integration, fusing together differing types of knowledge which were heretofore treated separately by separate tools. The integration of traditionally separate tasks (such as spreadsheet analysis, data management and program modeling) releases a previously untapped dimension of microcomputer power.

KNOWLEDGE INTEGRATION

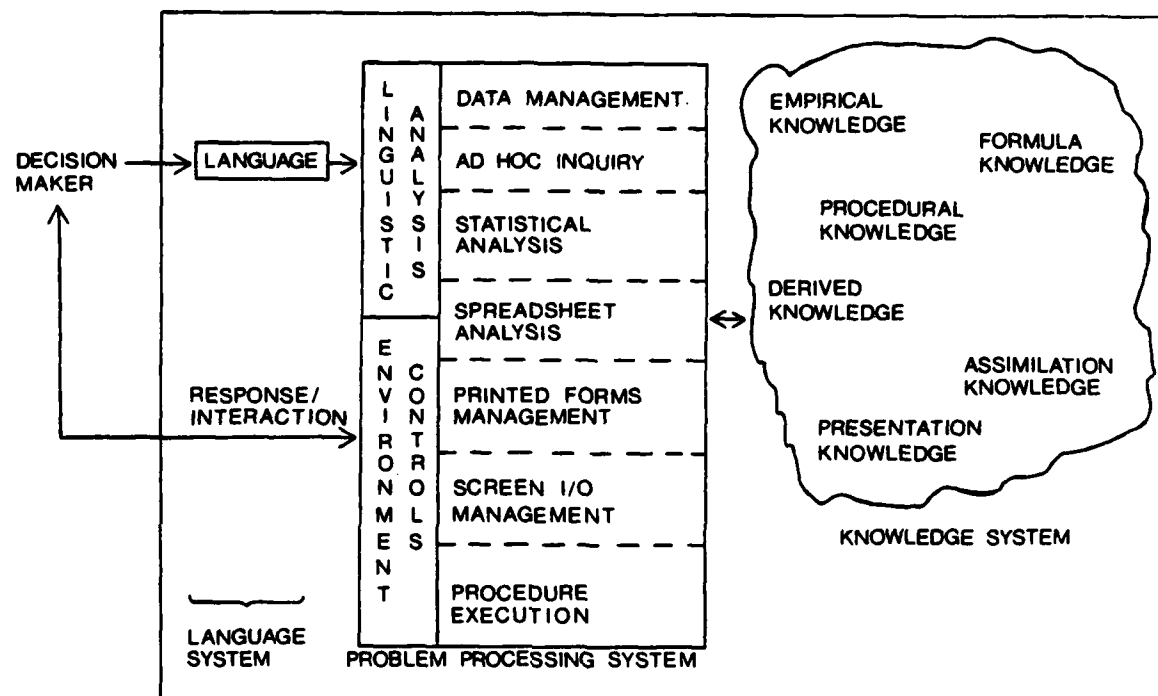
A decision maker utilizes several basic types of information or knowledge (2). There is a basic empirical knowledge about the realm within which the decision making takes place (e.g., the sales management world). This knowledge consists of observations about entities (e.g., customers, orders, products) that populate that world, their attributes, and their interrelationships. There is also formula knowledge which specifies how to derive new information from existing knowledge. Formula knowledge can also be used to derive speculations from hypothetical knowledge. Spreadsheet cell definitions are examples of formulas. Horn clause expressions and production rules (in the AI sense) are other types of formula knowledge.

Beyond formula knowledge, there is full-scale procedural knowledge, indicating an algorithm for deriving information. Empirical knowledge and knowledge derived from formulas and procedures typically need to be

presented in some manner. Thus, the decision maker may also be concerned with presentation knowledge, governing the way in which knowledge is presented. Conversely, there is knowledge pertaining to the system's assimilation of empirical and derived knowledge. This kind of knowledge controls the way in which (and the conditions under which) data is accepted as bona fide knowledge.

Effective integration of the various types of knowledge is accomplished when an assortment of piecemeal tools is replaced by a single integrated DSS development tool. This is a generic software tool for building decision support systems. All of the foregoing kinds of knowledge are stored in the knowledge system, where they are accessible to the generic DSS software as it responds to the decision maker's requests. This can be illustrated (2) as shown in Figure 2. Notice that in accordance with the framework represented in Figure 1, the DSS has two other elements, in addition to a knowledge system: a language system and a problem processing system.

A DSS user makes a request via the language system. The problem processor is the DSS software. It interprets the request and then solves the implied problem using its own inherent processing capabilities together with knowledge held in the knowledge system. These capabilities may range from data retrieval to model execution to model formulation (1). The result of a request may be a simple one-way response to the decision maker (e.g., the display of a report) or it may be an interactive session, in which the DSS user is prompted to supply additional information required by the problem processor in answering the request.



DECISION SUPPORT SYSTEM

Fig. 2 Concrete Structure of a DSS

A DSS language should be designed to minimize a decision maker's effort in stating a request. It should be non-procedural and English-like. The language should also be flexible enough to permit both non-procedural and procedural requests. The procedural capability enables the DSS user to specify his/her own customized procedures, beyond those existing in the knowledge system. As an added convenience, there should be a mechanism that enables a user to alter the language itself. Knowledge about language modifications should be held in the knowledge system to preserve the generality of the problem processor software. A modifiable language that allows both non-procedural and procedural, English-like requests implies a sophisticated

linguistic analysis component within the problem processing system, particularly where the DSS supports many types of information processing.

Beyond linguistic analysis, a problem processor performs various types of information processing tasks, such as spreadsheet analysis, procedure execution, data management, and so forth. The linguistic analysis may very well involve artificial intelligence techniques extending eventually to accepting user requests in a natural language. The PPS also is responsible for environment control. Environment control refers to the management of responses and interactions for a particular terminal type. Notice that the exercise of environment control depends on the availability of

environment knowledge. This is yet another kind of knowledge that could be held in the knowledge base.

Environment controls and linguistic analysis constitute the problem processing system's user interface. At the heart of a problem processor are its various information processing abilities. The variety of these abilities is important, but variety is by no means sufficient for a good DSS. There are four other crucial issues (2): suitable abilities, presence of indispensable abilities, extent of each ability, and integration of abilities.

Various types of problem processor abilities are particularly suitable for a decision maker. For managers and engineers, spreadsheet and statistical analysis capabilities are very suitable. Some abilities are indispensable. Generally, the data management ability is indispensable to a decision maker and modeling abilities (statistics, spreadsheets, procedures) are a close second. Clerical abilities (e.g., word processing) typically are not especially suitable for decision makers.

When assessing a problem processor, the extent of each ability should be carefully investigated. This is especially important for an indispensable ability. For instance, two problem processors may both have a data management ability. One may be very primitive and inflexible, while the other is highly sophisticated and quite powerful. Weak data handling cannot be compensated for by other abilities that are strong. For example, a fancy graphics facility is not particularly valuable when the underlying data used to generate graphs is insecure, inconsistent, or meager due to a weak data management ability. The

functionality of each problem processor ability (particularly the most important ones) should be at least comparable to the functionality of the better piecemeal packages targeted at the same task.

Integration of information management abilities is where the greatest return is realized. A problem processing system facilitating this capability has great potential. On the other hand, a problem processor may have several reasonably strong abilities, but without integration an enormous potential is lost. A non-integrated problem processor may do spreadsheet analysis or ad hoc inquiry or statistical analysis or procedure (i.e., model) execution in response to a request, but it cannot utilize two or more of these abilities simultaneously. It cannot integrate the various kinds of knowledge in the knowledge base (e.g., empirical, formula, procedural). This integration if it occurs at all, must be performed on an ad hoc basis by the DSS user or a DSS developer rather than being inherent in the DSS development system.

In contrast, a problem processing system whose abilities are integrated uses those abilities to handle knowledge integration automatically. Thus it can respond to more complex requests than a non-integrated PPS. Because the potential unleashed by integration is nearly limitless, we give only a few examples here. Traditional spreadsheet packages (e.g., VisiCalc, 1-2-3, etc.) make use of formula knowledge to carry out analyses, such as those needed in financial planning. When traditional spreadsheet analysis is integrated with procedure execution, any spreadsheet cell can be defined in terms of an entire program, rather than just a formula. Further integration with statistical analysis implies that a cell can be defined in

terms of statistics (e.g., standard deviation) derived from empirical knowledge, or in terms of a procedure that itself utilizes statistical analysis. Integration of spreadsheet analysis with an ad hoc inquiry ability means that cell values can be used to condition exploratory retrieval (or statistical analyses). Conversely, an ad hoc inquiry can be embedded within a procedure used to define a cell. Integration with a printed forms management ability allows results from ad hoc inquiries, spreadsheet cells, statistical analyses, and/or procedures to be blended together in a single customized report.

Ideally, a problem processing system should have indispensable abilities at the minimum and suitable abilities at the maximum. For the most important abilities, there should be no sacrifice in functionality relative to the best of the piecemeal packages. The abilities that are possessed should be integrated, allowing the problem processor to utilize diverse kinds of knowledge in response to a decision maker's request.

An extremely significant aspect of the DSS development system illustrated in Figure 2 is that its problem processor is general. Since all application-specific knowledge is held in the knowledge system, the PPS is not restricted to any particular application. That is, application-specific knowledge is not incorporated into the problem processor. As a result, the problem processor software is invariant to application changes. The importance of problem processor generality must be emphasized. It enables the same software tool to be used in a wide variety of application areas. When the nature of an application evolves, the only aspect of the DSS that changes is the content of its knowledge system.

THE KNOWLEDGE MANAGER

A general problem processor for a DSS based software development system is certainly a theoretically attractive and elegant notion, and it is consistent with the basic theory of generalized problem processing systems as introduced in the Foundations of Decision Support Systems (1). Nevertheless, we must ask whether such a processor is really feasible, and is it feasible for microcomputers?

In surveying existing micro software we can see the first primitive signs of a general problem processor in file management systems such as Condor (3) and dBase (4) which allow data management and ad hoc inquiry to be integrated with procedure execution. Other signs are MBA (5) and 1-2-3 (6) which are spreadsheet systems that have modest graphics abilities. Their "data management capabilities" are extremely limited relative to those of Condor and dBase. Beyond these, there is one software tool which is indeed a generalized problem processor, having all knowledge fusion capabilities cited earlier. This is KnowledgeMan (The Knowledge Manager) (7) which was recently introduced on 8086, 8088 micro computers under CP/M-86, MSDOS, and PC DOS.

The range of KnowledgeMan abilities covers all of the problem processor abilities shown in Figure 2. Thus KnowledgeMan is suitable for knowledge workers who are decision makers, planners, or researchers. It does not have a clerical orientation. To furnish an appreciation of the extent and integration of the seven abilities in KnowledgeMan, we shall briefly describe the highlights of each of them. This is by no means an exhaustive feature analysis.

Data Management

We begin with the indispensable ability of data management, which turns out to be very extensive in KnowledgeMan. Data management refers to the organization, creation, maintenance, retrieval, integrity, and security of data (i.e., empirical and derived knowledge in Figure 2). It is clear that the KnowledgeMan data management facilities were carefully designed to dominate those of common micro file handlers (e.g., Condor, dBase), in every respect. Thus the knowledge worker gives up nothing in using a general problem processor, rather than a more specialized data management tool. In fact, there are some significant gains, particularly in terms of capacity, security, efficiency, and ease of use.

KnowledgeMan enables data to be organized into tables of up to 255 fields apiece. For each field, a table has a column containing data values for that field. The table's rows therefore consist of data values (one for each field). Each row is called a record. For instance, the CUSTOMER table in Figure 3 has eight fields. Each row in the table is a customer record, containing information about a customer. The ORDER table has four fields and each of its rows pertains to a particular order. This tabular representation is appropriate for handling empirical and derived knowledge.

To define a table, the DSS developer simply specifies the desired name for the table (e.g., DEFINE CUSTOMER), and is then interactively prompted to

CUSTOMER	ID	FNAME	LNAME	STREET	CITY	STATE	ZIP	DATE
	53829	JOHN	DOE	100 MAIN	URBANA	IL	61801	821101
	67294	PAT	WEST	200 STATE	CHAMPAIGN	IL	61820	830216
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

ORDER	NUMBER	ID	DATE	AMOUNT
	123	67294	830301	259.02
	095	05039	830301	68.19
	281	53829	830302	5.23
	395	67294	830303	329.98
	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮

Fig. 3 Sample Tables

complete the definition. In the course of this interaction, the DSS developer specifies desired field names, field types (numeric, string, logical), field sizes (up to 65535 characters per field), and field pictures (for automatic editing and character-by-character integrity checking). Virtual fields can be defined using arbitrarily complex expressions; such fields are valuable because they guarantee integrity and utilize no storage space, while they can be manipulated just like actual fields. As a security measure, read and/or write access restrictions can be specified for the table and for any of its fields.

Once a table is defined, records can be created in that table. In the KnowledgeMan language there is a CREATE command, which causes the DSS developer to be interactively prompted to the new record's field values (e.g., CREATE CUSTOMER). Alternatively, there is another command which causes KnowledgeMan to load records from a file (external to the knowledge base) into a table in the knowledge base. Physically, all records created in a table are automatically encrypted by KnowledgeMan. This is a valuable data security provision, protecting data from casual viewing from the host operating system.

There are several provisions for maintenance, all of which are subject to write access privileges. At a very detailed level field values within a selected record can be changed one-by-one. Alternatively, a DSS user can browse through records, changing them as desired. A more global command allows all records meeting desired criteria to have their field values changed in some specified way. For instance,

CHANGE AMOUNT TO AMOUNT+3.50
FOR AMOUNT<100

causes a \$3.50 handling charge to be added to the order amount for all orders below \$100 (see Figure 3).

For extracting knowledge from a table, the DSS user has several options. Records can be extracted and displayed one-at-a-time, based on their positions in the table and whether their values meet desired conditions. To obtain the next customer record having a zip code of 61820 the command

OBTAIN NEXT FOR ZIP=61820

is used. Another way to extract a record is to make use of an index, which allows very fast access to a record based on a key value. For instance, ID might be an index key for CUSTOMER. To very rapidly extract the customer whose ID is 58342, the command

PLUCK 58342 FROM CUSTOMER

is stated by the knowledge worker. A key can be a composite of many expressions involving many fields. Micro file handlers that support indexing typically use only a sequential file index. KnowledgeMan uses the superior B+ tree indexing technique. Yet another type of extraction capability is provided by the high-level ad hoc inquiry language.

Ad Hoc Inquiry

Very often, DSS users need to undertake exploratory investigations of large volumes of knowledge. This might be viewed as a distillation process, selecting only those facts

which are pertinent to a particular decision or problem. Such explorations typically involve unanticipated, spur-of-the-moment inquiries. Ideally, the DSS user should be able to state a single English-like command which will extract the desired facts from one or more tables.

Such a facility does not exist in micro file handlers such as Condor and dBase. It does exist in MDBS III (8), the full-scale data base management system for micro computers, and in mainframe data base management systems such as IBM's SQL/DS (the de facto standard for the relational approach to data base management). The KnowledgeMan facility for ad hoc inquiry has a syntax that is very close to SQL/DS, though it does offer a couple of options not available in SQL/DS.

Suppose a DSS user desires a report of the name and identifier of each customer in Urbana who became a customer prior to September 15, 1982. The appropriate inquiry is

```
SELECT LNAME,FNAME,ID
FROM CUSTOMER WHERE
CITY="Urbana" & DATE<820915
```

If the result is desired in sorted format by (for instance) ascending last name: first name and descending zip code, then the query is

```
SELECT LNAME,FNAME,ID FROM
CUSTOMER

WHERE CITY="Urbana" &
DATE<820915

ORDER BY ASCENDING
LNAME,FNAME DESCENDING ZIP.
```

These examples operated only on one table.

The KnowledgeMan language is not limited to one table at a time, but can handle multiple tables simultaneously. For example,

```
SELECT
CUSTOMER.LNAME,NUMBER,AMOUNT

FROM ORDER WHERE
ORDER.DATE>830302

FROM CUSTOMER WHERE
CUSTOMER.ID=ORDER.ID
& ZIP IN (61801,61805)
```

produces a report of customer last name, together with order numbers and amounts, for orders after March 3, 1983 which were placed by a customer in zip code regions 61801 or 61805. To cause the report to be sorted by customer last name, with control breaks by customer ID, the following two clauses would be appended to the foregoing query.

```
GROUP BY CUSTOMER.ID
ORDER BY CUSTOMER.LNAME
```

At each control break (i.e., for each customer) KnowledgeMan computes and displays full statistics (including variance and standard deviation) for the customer's order amounts.

If desired, arbitrarily complex expressions can be used in a query. For instance,

```
SELECT UNIQUE ID, NUMBER,
(830307-DATE)*AMOUNT/1.395

FROM ORDER WHERE AMOUNT
SQRT(929.35+LOG(3.3))
ORDER BY ID .
```

KnowledgeMan makes all computations, including the application of various built-in functions such as square

root (SQRT) and logarithm (LOG). This example also illustrates the UNIQUE qualifier which can be applied to any selected field or expression. In this case, each ID value will appear only once in the output report regardless of how many order numbers are listed for that customer identifier.

Arbitrarily complex conditions can be stated within a KnowledgeMan query, using & (i.e., AND) OR, XOR (i.e., exclusive OR), and NOT logical operators. Wildcard symbol and string match conditions are supported. To extract all customers whose last names begin with M, or begin with Sm and end with th, the KnowledgeMan query is

```
SELECT ID,LNAME,FNAME FROM
      CUSTOMER
```

```
WHERE LNAME IN["M*","Sm$th"]
```

using * to indicate a wildcard string match and \$ to indicate a wildcard symbol match.

Statistical Analysis

Sometimes a DSS user may desire to gather statistics for field values in certain collections of records in one or more tables, without having all of the raw data displayed. KnowledgeMan users can accomplish this with a single, high-level command. Statistical computations will be performed not only on fields, but also on arbitrarily complex expressions. To limit the collection of records used in statistical computation, the DSS user can employ the same kinds of conditions used in ad hoc queries.

As an example, the following command computes full statistics for AMOUNT, and for each of the two expressions, using only those orders prior to March 15, 1983 placed by customers in Chicago.

```
STAT AMOUNT,
      ABS(SQRT(629.34)-AMOUNT),
      7.29+LOG(AMOUNT)

FROM ORDER WHERE DATE<830315

FROM CUSTOMER WHERE
      ORDER.ID=CUSTOMER.ID &
      CITY="Chicago"
```

Here ABS is a function that yields an absolute value. Three sets of statistics are produced. Each statistics set includes total, min, max, count, mean, variance, and standard deviation.

Not only are statistics displayed, they are also remembered by the system (in variables) for later use in other commands. For instance, the computed average of the last expression in the above STAT command is held in the variable #AVER(3) and the corresponding computed standard deviation is in #STDV(3). Later, we may want to pose an ad hoc query conditioned by these statistics:

```
SELECT NUMBER,ID,AMOUNT FROM
      ORDER

WHERE LOG(AMOUNT)+7.29 >
      #AVER(3)+2.0*#STDV(3) .
```

Notice that this enables the statistical analysis ability to be integrated with the ad hoc inquiry ability. Statistical variables can also be used in the course of data management. For example, a statistical variable can be used in defining a virtual field.

Spreadsheet Analysis

Spreadsheet analysis is a highly useful modeling facility for planners and decision makers, as demonstrated by the widespread popularity of packages such as VisiCalc (9) and SuperCalc (10). The KnowledgeMan spreadsheet ability offers features comparable to VisiCalc and SuperCalc. These include a 255 by 255 spreadsheet capacity, non-uniform column widths, automatic label spillover, viewing through a desired window, optional border suppression, cell definition replication (absolute and/or relative), printer output of cell values and/or cell definitions, cell pictures for automatic editing, lookup and summation functions, temporary override of cell values, and user controlled re-computation of cell values.

As with common spreadsheet packages, KnowledgeMan cells can be defined in terms of formulas. This formula knowledge is saved in the knowledge base. Formulas can be altered as desired. Statistical variables and fields can be used within cell formulas. However, the integration of spreadsheet activity with other KnowledgeMan facilities goes well beyond this. Each cell can be referenced as a variable (e.g., #C5 has the current value of the cell in the third column and fifth row). This means that cell variables can be used in data management, within expressions of ad hoc inquiries, and within statistical analysis commands. The following examples are valid statements:

```
CHANGE AMOUNT TO AMOUNT+#C5
  for AMOUNT<100
```

```
PLUCK #Z53 FROM CUSTOMER
```

```
SELECT
  ID,NUMBER,AMOUNT/SQRT(#LI3)
```

```
FROM ORDER WHERE DATE=#L2 OR
  DATE<#K2
STAT AMOUNT
  +LOG(#PI37*ABS(#PI36)) FROM
ORDER .
```

Non-spreadsheet commands such as these can be freely interspersed with spreadsheet commands.

An important innovation of the KnowledgeMan spreadsheet facility is that it allows a cell to be defined in terms of an entire procedure (i.e., program), rather than being restricted to simple formulas. As with formulas, procedural knowledge is held in the knowledge base. This integration of spreadsheets and programs is a radical departure from traditional capabilities and opens a new vista for DSS developers that need to organize the results of modeling in a spreadsheet form.

Procedure Execution

Many DSS users are confronted with problems that cannot be answered with spreadsheet or basic statistical analysis. Their analyses may require the use of complex algorithms (e.g., linear regression, proprietary forecasting techniques, optimization). These algorithms are specified in terms of programs. KnowledgeMan meets this type of modeling need by providing a comprehensive structured programming language. A procedure specified in this language can be written by the DSS developer and stored in the knowledge base and invoked at will by a DSS user. The procedure specification itself may have been devised either by the DSS developer or by someone with extensive modeling expertise who provides procedures to DSS developers as a service. The important point is that KnowledgeMan furnishes a facility for customized modeling.

Of the packages mentioned earlier, dBase offers a reasonable programming facility; Condor is considerably more limited; MBA and 1-2-3 have no appreciable facilities for customized modeling. However, dBase lacks many of the niceties that one would normally expect from a programming facility such as array processing, large numbers of working variables (i.e., more than only 64), test-case control structures, parameterized procedure execution, and unrestricted procedure nesting. All of these conveniences are supported for KnowledgeMan procedures. There are also the basics of assignment statements, if-then-else and while-do control structures (with no limit on nesting), step-wise procedure execution, and global/local declarations.

The integration of KnowledgeMan abilities means that fields, cells, and statistical variables can be treated just like any other variables within a procedure. Indefinite cell references are allowed, since KnowledgeMan permits a spreadsheet to be viewed as a large two-dimensional array. For instance, the cell to which $\#(I,J)$ refers is determined by the values of the variables I and J, which may themselves be all variables. Another aspect of the integration is that data management, ad hoc inquiry, and statistical analysis commands can be invoked whenever and wherever desired within a procedure. The command to enter the spreadsheet mode of processing can also be used within procedures.

As mentioned earlier, a cell can be defined in terms of an entire procedure. In fact, various cells in possibly different spreadsheets could be defined in terms of the same parcel of procedural knowledge held in the

knowledge base. For instance, a cell's value may be determined by a simulation program that uses data held in various tables or data derived from other procedures.

Printed Forms Management

Some DSS users are unconcerned about the precise layout of information on printed forms. For others, the generation of particular layouts that satisfy their presentation needs is important. Like many other software packages, KnowledgeMan provides a command that produces a line of output, beginning at a user-specified position. Repeated use of this command yields a desired printed report. However, KnowledgeMan also supplies a much more powerful, less cumbersome method for generating printed reports.

There exists a command for specifying characteristics of a printed form such as the form's name, positionings of titles and labels, mappings of variables and expressions to positions on the form, and the pictures to be used for editing the variable and expression values as they are printed. Each form declaration is held in the knowledge base as a part of the presentation knowledge which can be utilized by KnowledgeMan. A form declaration can be made by the knowledge worker or by someone else as a service to the knowledge worker.

A single command, indicating a particular form, is all that is needed to cause that form to be printed, complete with all present values of variables and expressions assigned to the form's declaration. All editing specified in the form declaration is automatically performed. The complete flexibility of positioning during form declaration means that

information can be printed appropriately on pre-printed forms; or entirely customized, non-pre-printed forms can be produced.

The integration of printed forms management with other KnowledgeMan abilities enables values of fields, cells, working variables, statistical variables, and/or expressions involving any of these, to be printed on a single form. Furthermore, the form print command can be interspersed with other commands and embedded within procedures. In the latter case, characteristics of the form (e.g., titles, labels, positions) can be altered in the knowledge base without affecting procedures that print that form.

Screen I/O Management

The foregoing abilities necessarily involve some kind of input and/or output through a console screen. When creating a record the DSS user is prompted for input. The opportunity to revise existing data is offered when browsing through a table's records. Output to the console screen can result from certain data management tasks, from ad hoc inquiry, statistical analysis, and spreadsheet processing. For the most part, these inputs and outputs follow certain standard formats dictated by KnowledgeMan. The screen I/O management ability gives a DSS developer the ability to depart from the standard formats.

At the most elemental level there are line-at-a-time screen input and output commands. The output works just like the line-at-a-time output to a printer, allowing the display of labels and expression values (with editing if desired) at a specified position. A line input command issues a prompt (if desired) at an indicated screen

position and accepts a data value entered by the user into a specified variable. For instance,

```
AT 11,16 INPUT FNAME WITH
    "First Name:"
```

displays the First Name prompt at the sixteenth position in row eleven. After the user enters a name, that name becomes the new value of the variable FNAME.

At a much higher level, a DSS user employs various commands, each of which processes an entire form-at-a-time. Screen forms are declared just as printed forms, except that input entries can be specified as well as outputs and various special effects can be specified. These special effects include any combination blinking, bells, underlining, and reverse video for any element (e.g., title, prompt, data value) in the screen form. Furthermore, the form can be partitioned into blocks, each with its own foreground color and background color. Screen forms constitute part of the knowledge base's presentation knowledge. They may have been declared by the knowledge worker or by someone else as a service.

The major screen form management commands are PUTFORM, GETFORM, TALLY, RESET and CLEAR. PUTFORM displays all output aspects of an indicated form. GETFORM accepts data input entries that have been declared for the form. If an entered data value is valid according to character-by-character integrity conditions declared for it, then it becomes the new value of a variable as indicated in the form declaration. TALLY re-evaluates each of a form's output expressions and displays the new values. Its effect is analogous to the recomputation of cell values

in a spreadsheet. RESET prepares the form to accept a new set of data entries and CLEAR clears an indicated form from the screen. Multiple forms (possibly overlapping) can simultaneously be on the console screen.

Screen form output can be declared as coming from fields, cells, working variables, statistical variables and/or expressions involving any of these. Conversely, the declaration can assign input values to fields, cells and/or working variables. Another aspect of the KnowledgeMan integration is that screen form I/O commands can be mixed at will with other commands and can be embedded within procedures. In the latter case, screen form positionings and special effects can be altered in the knowledge base without affecting procedures that utilize the form. Record creation and browsing can make use of customized screen forms, rather than the standard KnowledgeMan formats. Also a screen I/O form can be used as printed form, and vice versa.

With one exception, the screen forms management ability of KnowledgeMan is at least comparable to other micro screen handling facilities. The exception is Screen Master (II) which is an exhaustive screen management facility, typically used by professional application developers in conjunction with the post-relational data base management systems, MDBS III (8), for constructing very extensive micro application systems.

The User Interface

Aside from comprehending standard KnowledgeMan commands, the linguistic analysis component (recall Figure 2) allows DSS users to alter the

language to meet their own needs. This is accomplished by means of macro declarations. For convenience, a DSS user may want to declare that KnowledgeMan should recognize the word ADDRESS as being identical to STREET, CITY, STATE, ZIP. This is accomplished by

```
MACRO ADDRESS  
STREET,CITY,STATE,ZIP .
```

As a result, the query

```
SELECT ID,ADDRESS FROM  
CUSTOMER
```

is recognized as being identical to the query

```
SELECT ID,STREET,CITY,STATE,ZIP  
FROM CUSTOMER .
```

A macro can be declared as an alternative way of referring to any portion of a KnowledgeMan command. There is no size limit on macros and macros can be nested within macros to an arbitrary depth. Macro declarations can be held in the knowledge base as application-specific linguistic knowledge.

KnowledgeMan automatically controls its interaction with a DSS user according to the characteristics of the environment. For instance, KnowledgeMan uses a terminal driver table that is specific to the particular terminal being used. Also, a table showing which keys correspond to which control functions (e.g., destructive backspace) enables KnowledgeMan to conform to the environment. Finally, there are a couple dozen environment variables (e.g., printer depth) which also govern KnowledgeMan behavior during response and interaction. All of this environment knowledge can be held in the knowledge base.

APPLICATION DEVELOPMENT AND UTILITIES

A software tool such as KnowledgeMan can be used in two distinct modes. We have seen that it can be directly employed by a DSS user. It can also be used by a DSS application developer, who develops an application software package (or turn-key system) by prespecifying application-specific table definitions, formulas, procedures, forms, and macros. Also the terminal driver table, control function key assignments, and environment variables can be pre-stored in the knowledge base. This would enable the developer to present a DSS user with a system built from KnowledgeMan and an application-specific knowledge base, yet the DSS user may be entirely unfamiliar with KnowledgeMan itself. However, as the DSS user's needs outgrow the capabilities of the application system, the end user can begin to employ the KnowledgeMan abilities directly since KnowledgeMan is already in place.

There is also a KnowledgeMan facility that allows the tabular portions of its knowledge base to be readily useable to independent application systems and utilities (e.g., graphics packages). With a single command, the knowledge worker can cause KnowledgeMan to selectively extract data from one or more tables and deposit the result in an operating system file that has either a DIF, ASCII, or BASIC-compatible format. All ad hoc inquiry features (except control breaks) are available with this conversion command. The net effect is that any application system or general utility, which utilizes files with one of these kinds of formats, is readily compatible with KnowledgeMan as an add-on or back-end processor.

The KnowledgeMan facility for loading batches of ASCII records with a single command opens yet another avenue for KnowledgeMan usage. It enables data that has been extracted from large scale data bases to be readily incorporated into a local knowledge base, for personal use by a knowledge worker. The large scale data base may reside on a mainframe, mini, or micro. As an example, large scale integrated application systems are built for micros using MDBS III. A single MDBS query is all that is needed to generate a file of data from a large scale integrated data base. This file can be immediately loaded into a KnowledgeMan table by a single KnowledgeMan command, without any intermediate file processing. The net effect is that many kinds of problem solving can be easily offloaded from an organization's integrated data base, for localized use by a decision maker. Figure 4 illustrates a typical configuration.

CONCLUSION

We have examined how software can contribute to the objectives of knowledge fusion. In particular, an architecture for knowledge management systems based on a general problem processor was introduced. The important characteristics of a general problem processor were identified. These traits are consistent with decision support systems theory (1). At first glance, one might question the feasibility of such an ambitious piece of software, particularly on micros. However, the feasibility of an extensive general problem processor for micros is proven by the existence of KnowledgeMan.

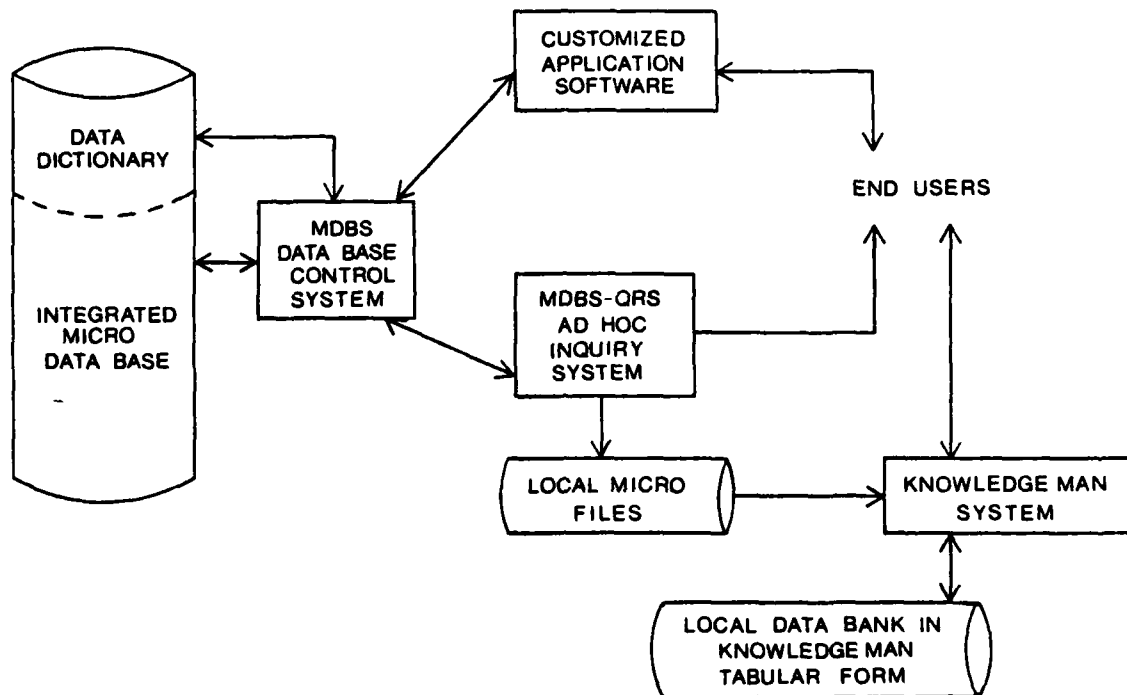


Fig. 4 Integrated Application System on a Micro

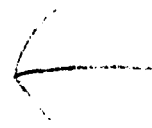
It is interesting that, from a global perspective, KnowledgeMan displays a high degree of artificial intelligence. The summary description of KnowledgeMan features illustrates what is meant by the variety, suitability, indispensability, extensiveness, and integration of knowledge management abilities within a general pro-

blem processor. While this brief summary barely begins to do justice to the KnowledgeMan capabilities, it is suggestive of the flexibility and power we should expect from a software tool for knowledge integration. We expect that this breakthrough in knowledge fusion will stimulate further research in the realm of generalized problem

processing systems and eventually lead to the implementation of tools with even greater capabilities than KnowledgeMan.

REFERENCES

1. R.H. Bonczek, C.W. Holsapple and A.B. Whinston, Foundations of Decision Support Systems (forwarded by Herbert Simon), Academic Press, New York, 1981.
2. C.W. Holsapple and A.B. Whinston, "Software Tools for Knowledge Fusion," Computer-world -InDepth, April 11, 1983.
3. Condor User Manual, Condor Computer Corp., Ann Arbor, Michigan.
4. dBase User Manual, Ashton-Tate, Culver City, California.
5. MBA User Manual, Context Management Systems Corp., Torrance, California.
6. 1-2-3 User Manual, Lotus Development Corp., Cambridge, Massachusetts.
7. KnowledgeMan User Manual, Micro Data Base Systems, Inc., Lafayette, Indiana.
8. MDBS III User Manual, Micro Data Base Systems, Inc., Lafayette, Indiana.
9. VisiCalc User Manual, VisiCorp, San Jose, California.
10. SuperCalc User Manual, Sorcim Corp., Santa Clara, California.
11. Screen Master User Manual, Micro Data Base Systems, Inc., Lafayette, Indiana.



APPLICATION OF ARTIFICIAL INTELLIGENCE TO TACTICAL OPERATIONS

Major Timothy Campen
HQ Joint Special Operations Command

Don E. Gordon
HRB-Singer, Inc.

If a next world war is fought, the best use of knowledge, not only facts, will be as determinant to victory as was the use of Ultra in WW II. The United States had best have supremacy in the high-technology area, especially with regard to the high-speed heuristic handling of data and specifically using data to develop knowledge - to develop intelligence.

Numerical superiority in weapons and manpower will enable enemy forces to deploy along the entire border and at the same time to concentrate major combat power for breakthrough operations in the area and at the time of their choice. The biggest threat is not a ten-foot-tall enemy; the biggest threat is that NATO may lose key first battles because it has too much data.

More information will be collected for the battle than ever before. Both opponents will be confronted with handling unsurpassed quantities of information to use for operational planning and intelligence. The force that can get the information needed the most and use it the best will have an advantage far more critical than numerical superiority of combat forces.

If there is an attack, we can expect it along our entire defensive line simultaneously, an offensive in which enemy tank armies attacking with two tank divisions will attempt breakthrough operations across a front

8 to 15 kms wide. They attack in echelons. Once the first echelon is deployed, other divisions follow to maintain the momentum of the attack.

To continue the example, a U.S. armored division is deployed to defend a front at least 30 kms wide with about 300 tanks, and a host of anti-tank weapons. But, they'll be outnumbered by a ratio of 6:1 in tanks, 2,000 enemy tanks. The U.S. commander will confront about 100 artillery tubes along each kilometer of the breakthrough corridor. The U.S. commander will have to trade space for time -- he will have to defend before he can conduct the deep attacks called for in FM 100-5 before he can fight the Airland Battle.

Enemy tactical doctrine is based on the strength of the offensive as the most decisive form of combat. In conventional combat, this concept will be characterized by:

- Attacks by massed tank and motorized (mechanized) units,
- High rates of advance (30-50 kms a day) on the main axis,
- Movement by day and night and in all weather, and if necessary, during the conduct of NBC operations,
- Tactical surprise,

- Support of the ground attack with strong tactical air forces to achieve air superiority at points of their choosing and for the conduct of deep air strikes,
- Strong mobile anti-aircraft protection,
- Airborne and helicopter operations in depth,
- Massive artillery support, and
- Radioelectronic combat (electronic warfare) used as a weapon system.

There is nothing new in that assessment, it continues to stand the test of time. The enemy can do this and they can probably do it without dependence on elaborate electronic systems, substituting instead rigid planning and an acceptance of high casualties. To do this with economy of force, the Soviets must first determine where best to attack, the enemy too must first process a lot of data and a lot of knowledge.

The U.S. Army has tactics to confront this threat. These tactics require that generals and colonels move highly mobile fighting units, battalions and companies to key battlefield positions at the best time and place to fight the enemy forces making a penetration and to counter the enemy offensive with deep-penetrating counterattacks by highly mobile forces.

- To do this, commanders of fighting units must see and engage the enemy at the maximum effective range.
- Brigade and battalion commanders must slow the enemy rate of movement.

- Division commanders must reinforce as rapidly as possible at the most decisive point.
- The defense must be kept flexible and elastic. Units in forward battle positions must be moved to other positions on the flanks and to the rear after they have accomplished maximum attrition, or to conduct penetrating attacks deep in the enemy's rear area. This is the most challenging aspect of the active defense and depends on both a superb communication and information processing system to direct minute-by-minute tactical instruction required by the commander.
- The commander must effectively use the trilogy of fire, maneuver and EW as a weapons system. Most importantly, the commander must select the most needed information to carefully choose the most important enemy targets, he must not become overwhelmed by information at a time when intelligence collectors can easily flood analysis and processing systems. The most successful enemy counterintelligence techniques may not be to deprive U.S. forces of intelligence, but to flood our intelligence systems with overwhelming valid data.

The use of artificial intelligence expert systems are needed not only to process and analyze unprecedented amounts of intelligence data, but also to filter the critical from the noncritical data and to expedite the intelligence needed the most by commanders and decision-makers who can use it the best. An intelligence expert system is needed to substitute

for the present-day intelligence data bases used by one intelligence function to feed another intelligence function. The problem with intelligence data processing today is that the systems are designed for the wrong customers. Intelligence data bases are not designed for operators -- commanders and operations officers -- they are designed for other intelligence users. The entire Washington intelligence system linking DIA, NSA, and CIA is designed for those agencies to exchange data. Needed are systems that feed commanders and operators.

While considerable advances have been made in designing expert systems in the fields of medicine, oil and mineral exploration, and terrain evaluation, probably no other field lends itself to the design of a heuristic expert system better than does military operations. There are direct correlations between the heuristic evaluations used in gaming strategy involving tic-tac-toe, chess and military tactics.

Computerized tic-tac-toe serves as an example by which a method referred to as "minimax" -- "Min" and "Max" being opponents -- may be used in which available moves (values) for "Max" are subtracted from potential moves (values) for "Min". This is an exhaustive search procedure where each possible move is evaluated within the context of all possible follow-on moves.

When applied to tic-tac-toe with a depth of two, the range of possible moves looks like those shown in Figure 1.

When extended to a depth of nine, there are 15,000 possible moves in this simple game. And when this procedure is applied to a more complex game like chess, exhaustive modeling yields a staggering

combinational explosion of 10^{120} possible moves. In order to avoid such a vast and intractable range of choices, called a combinational explosion, heuristics (or rules of good judgment and experience) must be applied. Returning to our example of tic-tac-toe and applying heuristics to the game, we need not consider all of the 15,000 or 10^{120} possible moves. Strict adherence to two simple heuristically derived rules prevents a combinational explosion.

First move for each player should be to the center or to a corner.

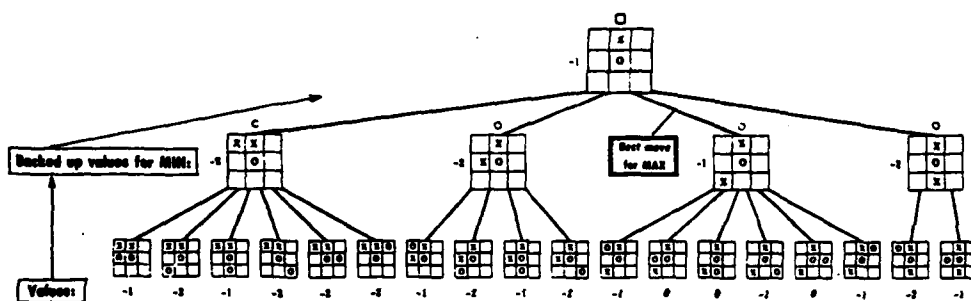
Subsequent moves should be to win or block if necessary.

This will insure at least a draw in nine moves. We see by this simple example (see Figure 2) that the use of well chosen pattern sets (heuristics) can help us maintain control over otherwise unmanageable exhaustive searches.

Infinitely more involved and varied than chess, the decision tree for a major military operation would, without the use of heuristics, result in a combinational explosion of such magnitude as to defy reasonable search times. (See Figure 3)

What follows is a very simple example of how the use of artificial intelligence could be applied to the battle -- fighting needs of a commander. The example is not intended to demonstrate or even approximate the full capability of artificial intelligence. The model chosen (Figure 4) provides the perspective of a Soviet combined arms army commander attempting a penetration against NATO forces in Europe. The Soviet system which is totally fabricated was selected to show application of an artificial

THE MINIMAX PROCEDURE



○
DRIVE A

○
DRIVE B

○
DEFAULT

VULGA IIe



Figure 1

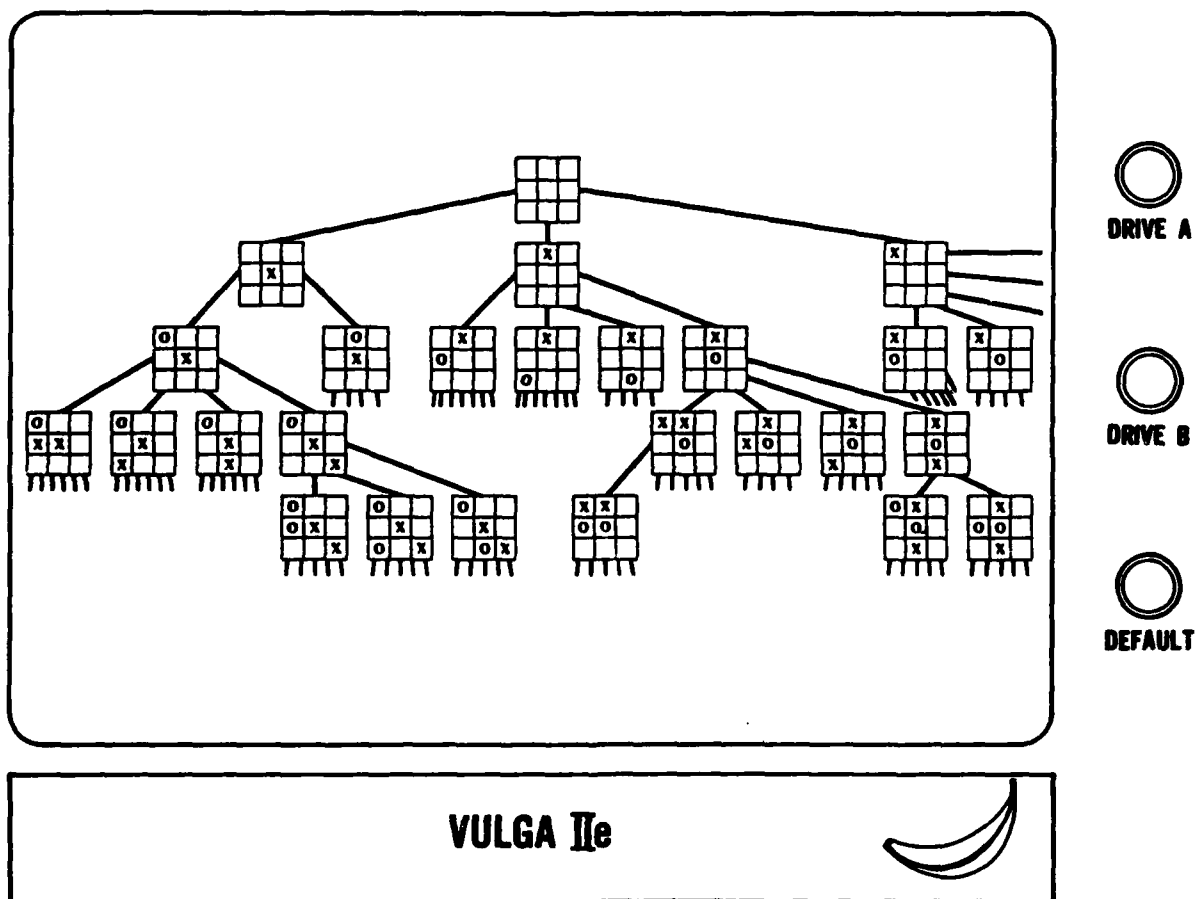


Figure 2

LIMITATION OF MINIMAX:

WHEN THE COMPUTER IS ON THE LOSING SIDE IT EXERTS ITS POWER AT DELAYING CATASTROPHIES BEYOND THE LIMITED HORIZON WHICH IS IMPOSED BY THE DEPTH OF SEARCH, RATHER THAN PREVENTING IMMEDIATE LOSSES.

Figure 3

THE IDENTITY OF ENEMY UNIT LOCATED VIC QE 260890 THE STATUS OF THIS UNIT IS: 93% PROBABILITY THAT THIS UNIT IS 1-37 MECH INF, 7 I.D. USA. 4% PROBABILITY THAT THIS UNIT IS 1-86 MECH INF, 7 I.D. USA. 3% PROBABILITY THAT THIS UNIT IS UNIDENTIFIED. THE UNIT'S COMMAND POST LOCATED 50 METERS CEP QE 26458995. THE UNIT CONDUCTING A DELAY, WHICH IS RATIONAL.	☐ DRIVE A
	☐ DRIVE B
	☐ DEFAULT

VULGA IIe	

Figure 4

intelligence system and to avoid a classified presentation. The Soviet commander is attempting multiple penetrations across his entire front. His mission --secure a specific objective 75 kilometers to our rear. His essential element of information, --his EEI: --where is the weakest NATO defense point along his combined army's front? Without going into a detailed explanation of tactics, let it suffice that it is important to the Soviet commander to determine where a NATO force is delaying and where it is defending. Under most circumstances it is preferable to attack a delaying rather than a defending force. The Soviet commander riding in his armored BMP-Variant command and control vehicle knows this and turns to his multiple-drive, double-sided, double-density disk VULGA IIE microcomputer (distinguished by the banana logo on the front panel) and calls up the GRU's NATO Focus Expert System on the amber tinted screen. NATO Focus is his artificial intelligence software. He enters the initial query: identify enemy unit: LOC VIC QE 260890. Give status. The machine whirs, a small red light glows indicating disk drives in operation while a blinking amber light indicates a data exchange with the supporting GRU mainframe at the 7th Guards Army HQs to the rear. Almost instantaneously, three green lights glow and the following data is displayed on the screen in front of the Soviet commander.

The screen indicates the probability of enemy unit identity, the unit's location, and that the unit is most likely conducting a delay, a performance the artificial intelligence system judges to be rational from the perspective of the U.S. commander based on all information available at the GRU's mainframe. Not yet comfortable with heuristic functioning, the Soviet

commander challenges for more proof. He hits the challenge key...Why 93% prob? The response is immediate. (See Figure 5).

Before the Soviet commander can punch the desist key, the amber screen pours forth even more data confirming the VULGA's heuristic conclusion as though in retribution against the questioning commander. The computer is calculating, screening, and evaluating thousands of pieces of data in nanoseconds. (See Figure 6)

Satisfied that he knows which unit is to his front, he challenges the VULGA to support its conclusion that the I-37 MECH INF is in fact delaying. He types in: Confirm status I-37 MECH INF delay VIC QE 260890. The VULGA about wrenches itself free from its mounting bolts and hydraulic shock absorbing pillows as it flashes a pattern of individually numbered multi-colored circles on the screen. (See Figure 7)

The computer graphic circles represent information reported from collectors on the battlefield. Some of these circles are fully colored meaning that intelligence indicating activity in that specific functional area has been received and processed. The absence of color means that no monitored activity has been reported relating to that tactic or characteristic. Each circle in the display represents either a discrete action step of a doctrinal characteristic, called a preincident indicator, of a U.S. force conducting a delay. The numbers associated with each circle are keyed to event descriptors shown on these displays. (See Figure 8)

The VULGA IIE computer is capable of displaying all four of the following menus simultaneously on the same screen. Displayed before the Soviet

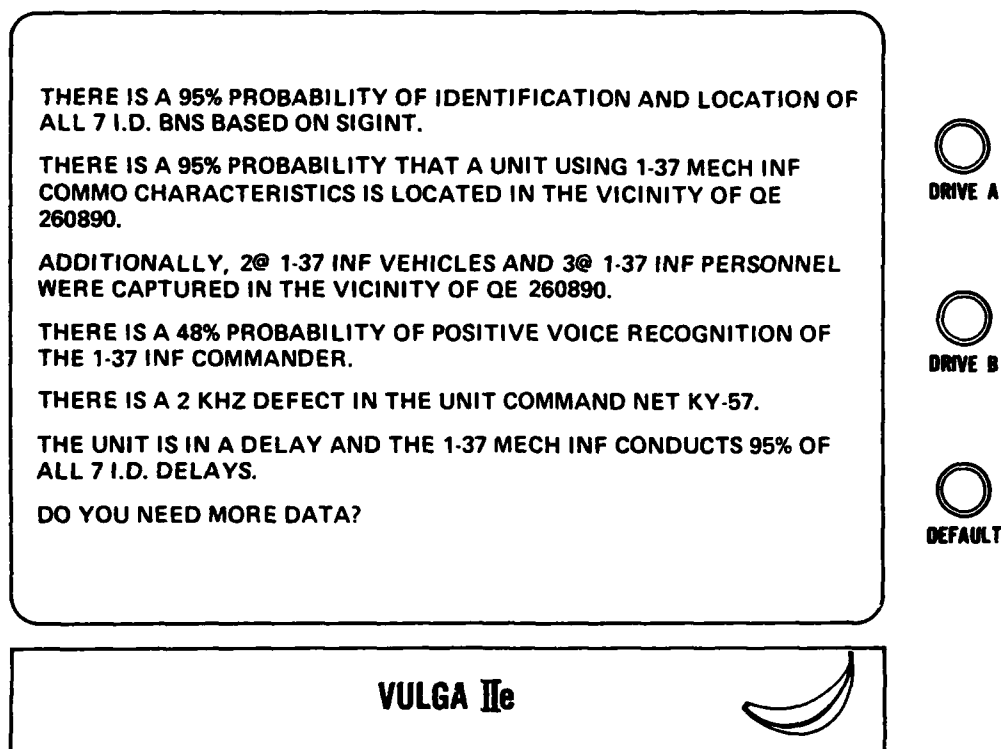


Figure 5

commander is a list broken down into four categories describing the characteristics of a delaying force according to U.S. Army literature, and finally heuristic rules which judge rational reasons for the conduct of a delay. (See Figure 9)

These lists separate all the factors that form the heuristic rules pertaining to the U.S. version of the conduct of a delay from alternate positions. (See Figure 10)

The factors and rules were compiled after reading U.S. and NATO doctrinal literature, from reading after action reports, a wealth of information available in the professional literature and by collecting volumes of information from radio monitoring,

satellite photography and observing first-hand NATO exercises and training. When operant deviations to published doctrine were detected in exercises and training, they were loaded as empirical data and as alternatives, variations or modifications. Doctrine and its operant variations remained as clear as the boundaries in a tic-tac-toe or chess game.

Meanwhile, the U.S. commander of the 1st Brigade perceived his unit's activity in four dimensions. The 1-37th MECH INF was organizing the initial and secondary delay positions at locations A and B. The 1-37th MECH INF reinforced with another mechanized rifle company and a tank company. (See Figure 11)

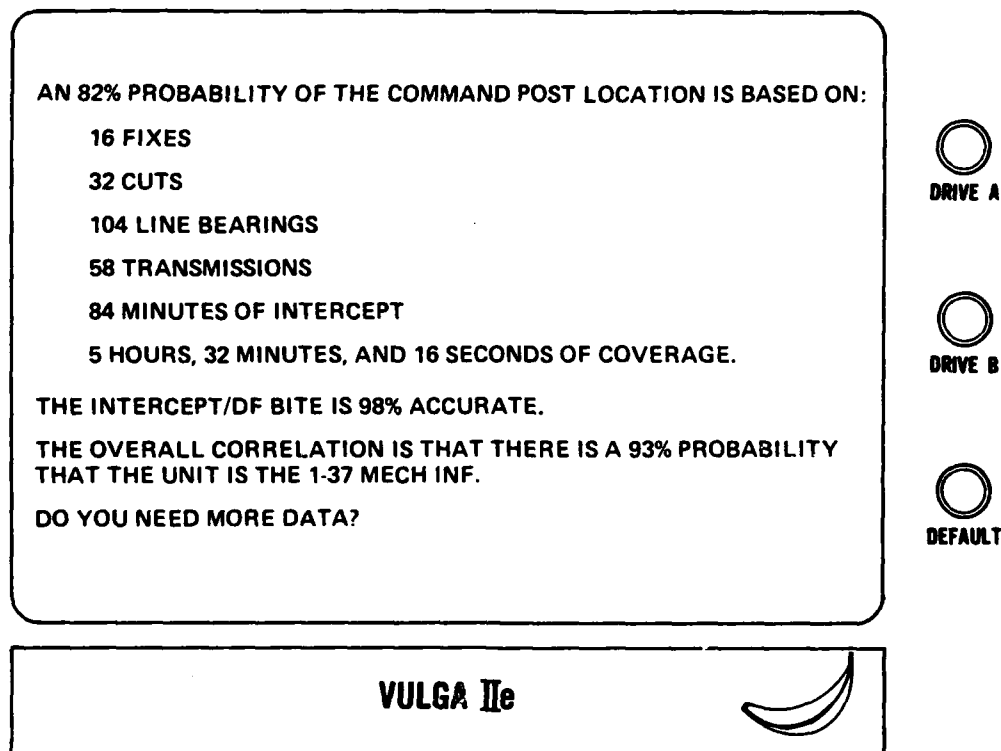


Figure 6

Elements from the initial position delayed back through the second position to the third position. (See Figure 12)

Elements at the second position picked up the delay while the third delay position was occupied. All this time jammers from the divisional CEWI Battalion maintained a one-kilometer distance from each delaying position while jamming (obvious by their excessive power) enemy communications. Corps provided priority of fires to the battalion task force using a corps forward observer (with his peculiar radio signature) accompanying the task force commander. During the delay, the task force maintained contact with the Soviet forces. Everything the U.S. commander did was announced by his electronic systems. While secure communication provided voice and data security, that very method

provided vital clues. When the U.S. commander needed communication most, it was useless because his omnidirectional antennas were no match for Soviet jamming. High technology support to the 1-37th task force commander was limited to his digital watch and tacfire. The 7th ID which did have a computer was using its van-mounted IBM-407 or IBM-360 card processing systems to correct last month's pay vouchers.

The NATO Focus Expert system used by the Soviet commander, meanwhile, consumed large quantities of intelligence from SIGINT, ground reconnaissance, airborne platforms and contact reports. The GRU intelligence unit assigned to the 7th guards army did not rely on unit intelligence officers to dribble back reports. Instead GRU officers accompanied every reconnaissance unit, every combat unit, and

AD-A139 685

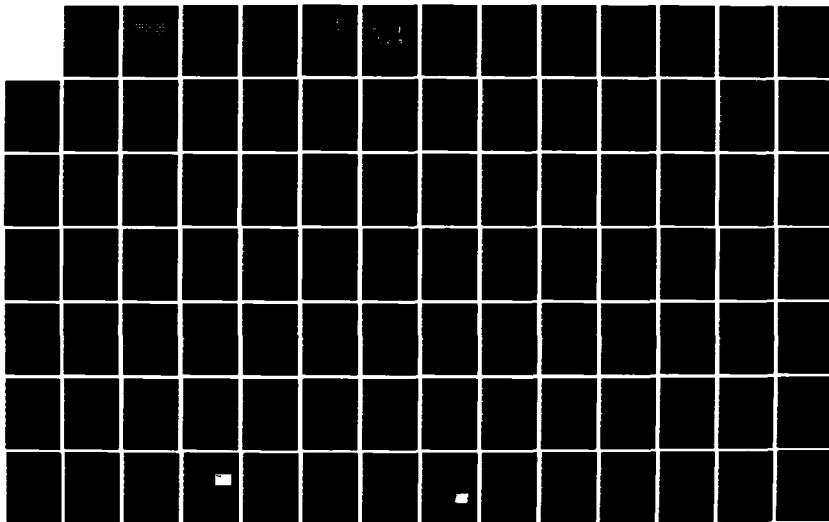
PROCEEDINGS OF THE ARMY CONFERENCE ON APPLICATION OF
ARTIFICIAL INTELLIGENCE (U) BATTELLE WASHINGTON
OPERATIONS DC B J TULLINGTON 31 JAN 84

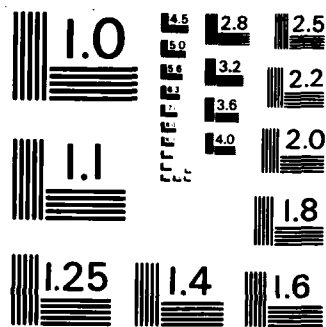
24

UNCLASSIFIED DARG29-81-D-0100

F/G 5/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

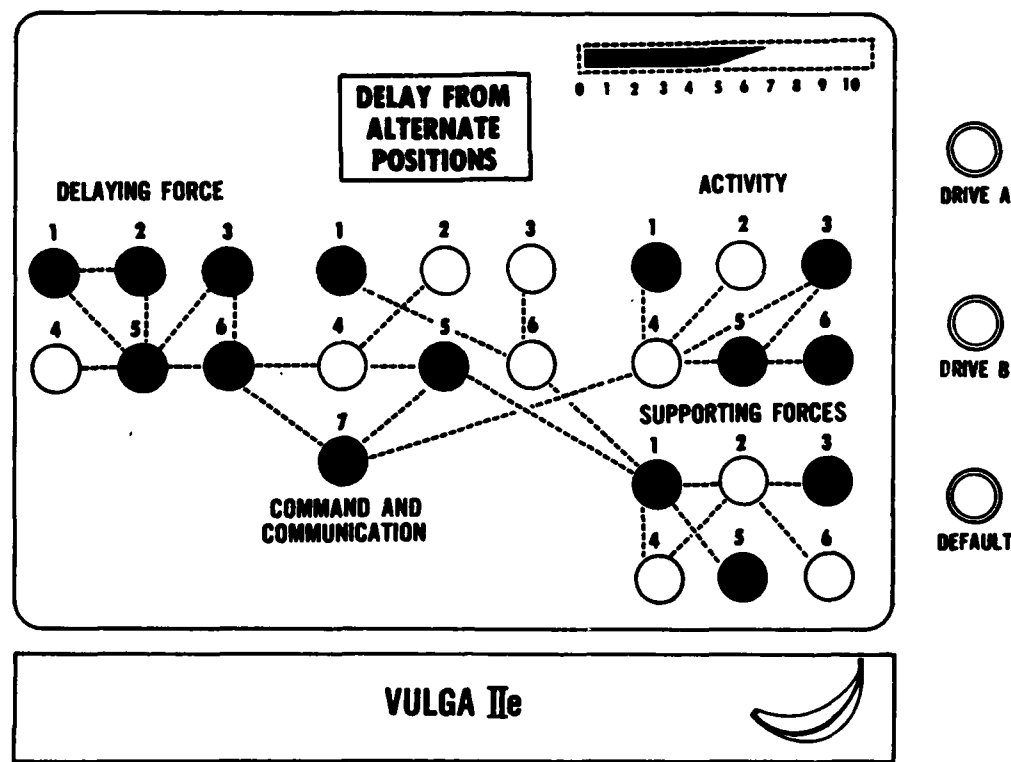


Figure 7

were organic to SIGINT and airborne collectors communicating directly with the NATO Focus Expert system. Every bit of data that entered the system was evaluated based upon a heuristically determined model of U.S. tactics and doctrine. A portion of that model and its decision tree is shown in Figure 13.

NATO Focus accepted that the U.S. 7th ID could perform only two types of operations when in contact. The 7th ID could conduct only offensive or defensive operations. There are only three offensive and three defensive operations. Each offensive or defensive operation had one or more, but a limited number of, variations. Each variation could be conducted by a given number of methods. As the

heuristics system within NATO Focus was evaluating these factors, several important clues were reported by front line Soviet collectors. Reconnaissance parties reported that hastily fallen trees covered with concertina wire were blocking several roads in the vicinity of an unidentified battalion at QE 260890. Intercepted radio traffic made several references to task force and armor augmentation of a mechanized unit in this same area. These are but two examples of an overwhelming number of sensor and collection reports that were parsed and fed to the commander at the right time. When processed through the heuristic rule-based system, this intelligence was found to be a clear doctrinal and empirical indication of a defensive operation. (See Figure 14)

MENU	
DELAYING FORCE 1. HIGHLY MOBILE 2. ARMOR HEAVY 3. TASK ORGANIZED 4. TWO MANEUVER UNITS 5. FIRST UNIT PASSES THROUGH SECOND 6. SIMPLE ORGANIZATION	ACTIVITY 1. SHORT/VIOLENT COUNTERATTACKS 2. AMBUSHES 3. AGGRESSIVE RECON 4. LONG RANGE FIRES 5. USES NATURAL OBSTACLES 6. HARASSING ATTACKS
COMMAND AND COMMUNICATION 1. MULTIPLE VHF NONSECURE NETS 2. ONE NCS - NONSECURE 3. OUTSTATION ON DIV AND CORPS NETS - SECURE 4. LONG LINK DISTANCES 5. SIMPLE CONTROLS 6. DIRECT VHF/HF WITH CEWI 7. DIV AND CORPS FO NETS	SUPPORTING FORCES 1. PRIORITY OF CORPS FIRES 2. JAMMERS DEPLOYED WELL FORWARD 3. HELICOPTERS IN RESERVE, DS TO TF CMDR 4. ENGINEERS IN DEPTH 5. REINFORCING ARTY FIRES TO BN TF 6. VERY LIMITED IF ANY ADA


DRIVE A


DRIVE B


DEFAULT

VULGA IIe 

Figure 8

DELAY
MAINTAIN CONTACT WITH ENEMY CAUSE ENEMY TO CONDUCT SUCCESSIVE ATTACKS GAIN TIME COVER DEPLOYMENT OF MOVEMENT RETIREMENT WITHDRAWAL RETREAT

VULGA IIe 


DRIVE A


DRIVE B


DEFAULT

Figure 9

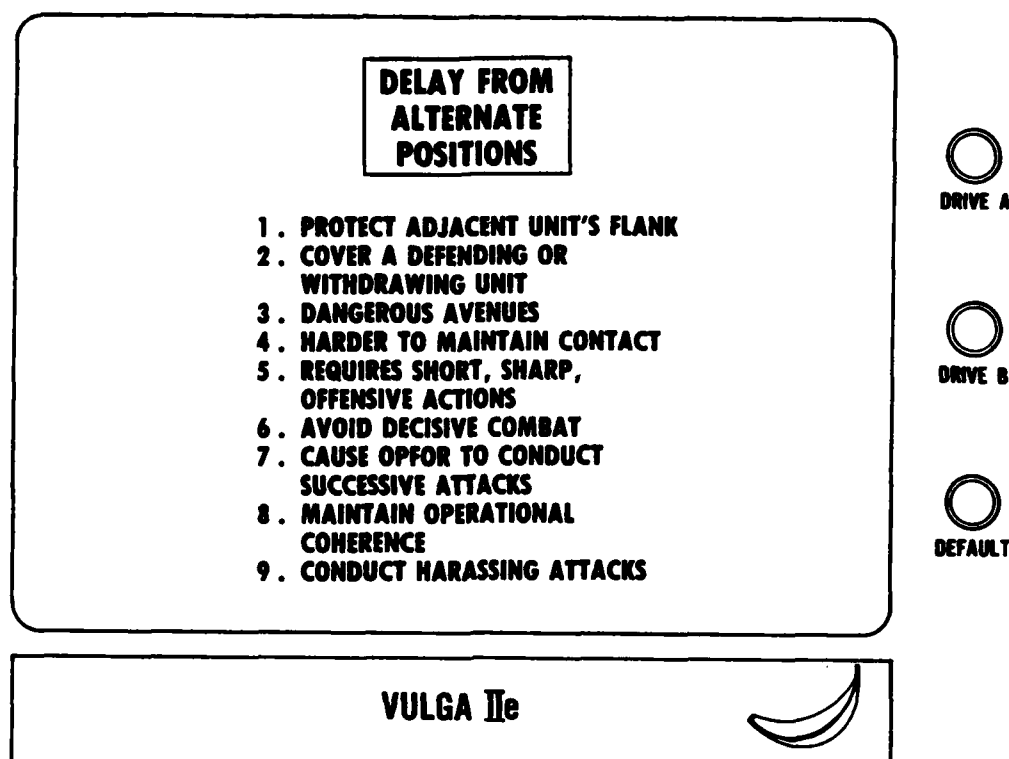


Figure 10

Additional information indicated a retrograde type defensive operation. Selecting next between protection, withdrawal, or delay, heuristics were used to conclude that the unit was conducting a retrograde and specifically a delay. (See Figures 15 and 16)

As contact was continued, heuristic selection, unit identification, and a wide variety of tactical information indicated that a task force controlled by the 1-37 MECH INF for the 1st Brigade was conducting a delay from alternate, rather than successive positions, though a combination of both was most likely the actual course. NATO Focus consolidated all that data, combined it with knowledge to mimic human analysis, and most importantly, tested its conclusions against negative hypothesis. (See Figure 17)

Heuristics determined that information shown in Figure 18, indicators of other type operations was not present or if present, was qualified. It also applied logic to determine if it was rational for the 1-37 MECH INF to delay.

NATO Focus determined probability. It did all this not only for the Soviet commander confronting the 1-37th MECH INF task force, but for all Soviet commanders attacking along the front. NATO Focus presented similar data, probably with lower probabilities of accuracy, on many company-size units, all NATO battalions, brigades, regiments, divisions, corps, and most special units along the entire army front. Artificial intelligence did not replace all the Soviet analysts, it did not replace a single analyst. AI did increase the productivity of

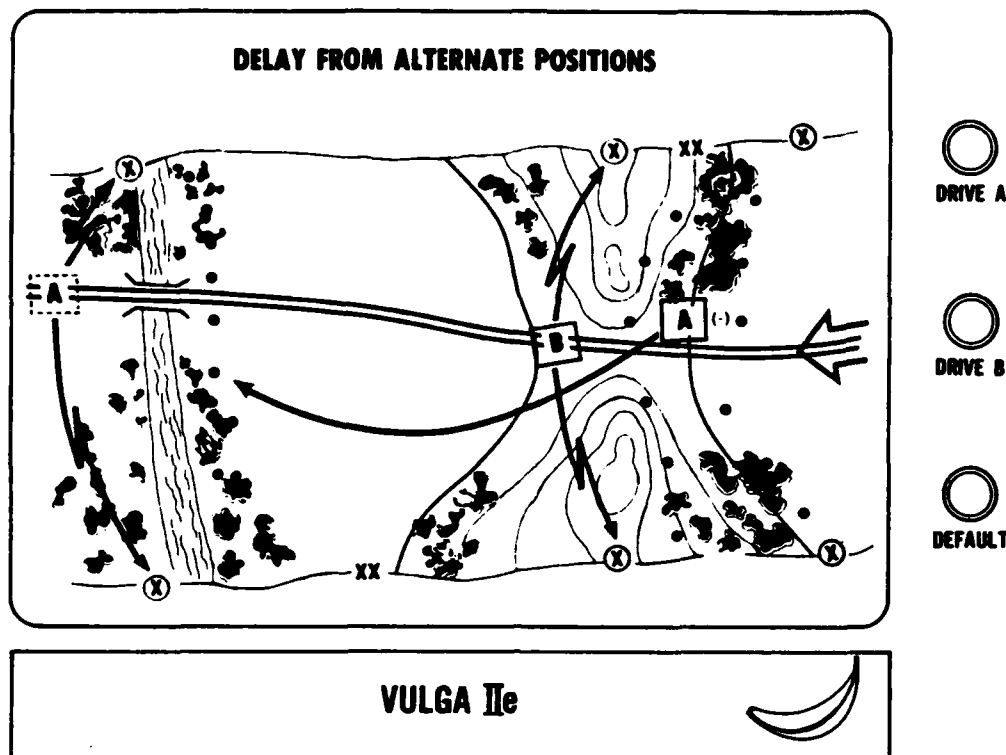


Figure 11

the best analysts and helped inexperienced or poor analysts to perform as well as the best analysts had previously performed. NATO Focus prevented the system from being flooded by trivia that had no impact on winning while selecting critical detail that was vital. Those empty circles discussed earlier indicated intelligence gaps and helped to direct new collection priorities.

The above is just a tickle, intended to stimulate thinking and future requirements. Similar military systems within DOD for different functions are already in partial use with advanced design and operation scheduled for completion by 1985. The proposed

issue to all students at the U.S. Army's Command and General Staff College of personal type computers for temporary use during the academic year is a step in the right direction. We have much further to go and we need to go much faster if this nation and its armed forces are to maintain its technological advantage and superiority.

A laundry list of proposed computer policy is not offered, nor appropriate. Policy is the antithesis of discovery. Find money wherever possible and use it to discover, leave the caution, policy, and regulations to those who make a profession of history. (See Figure 19)

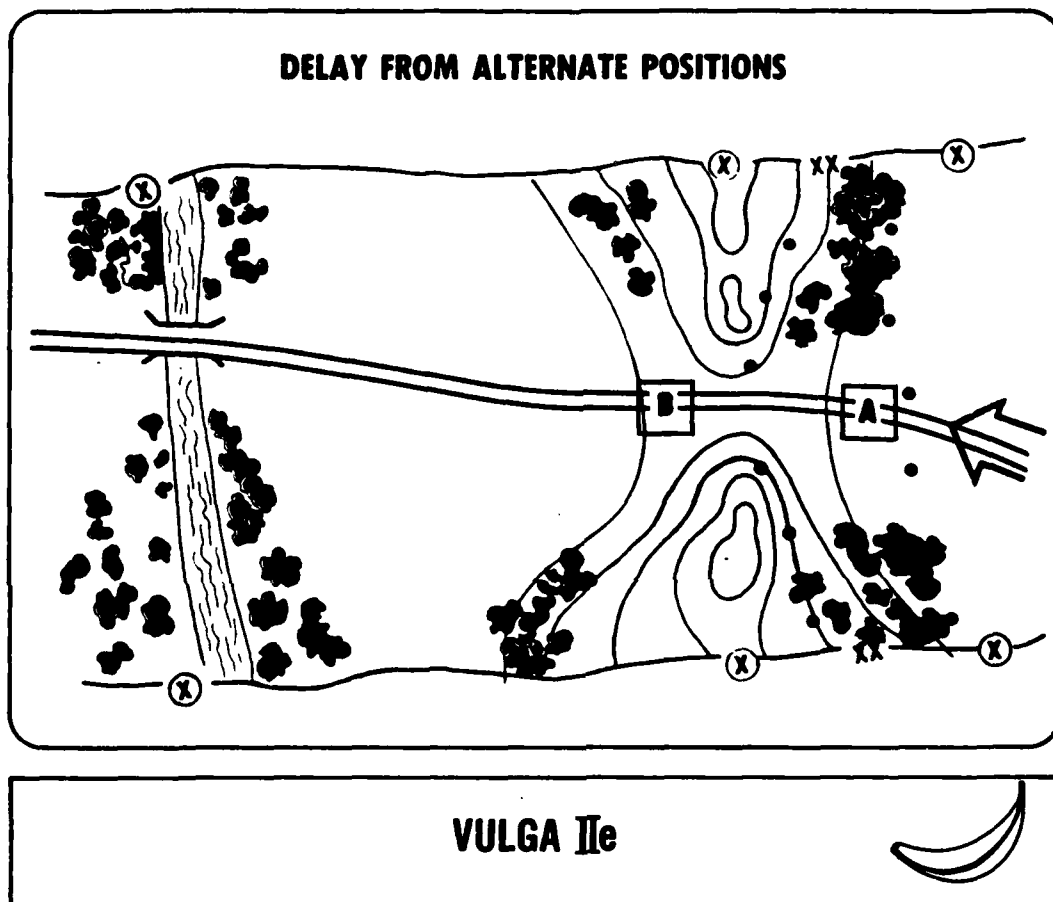


Figure 12

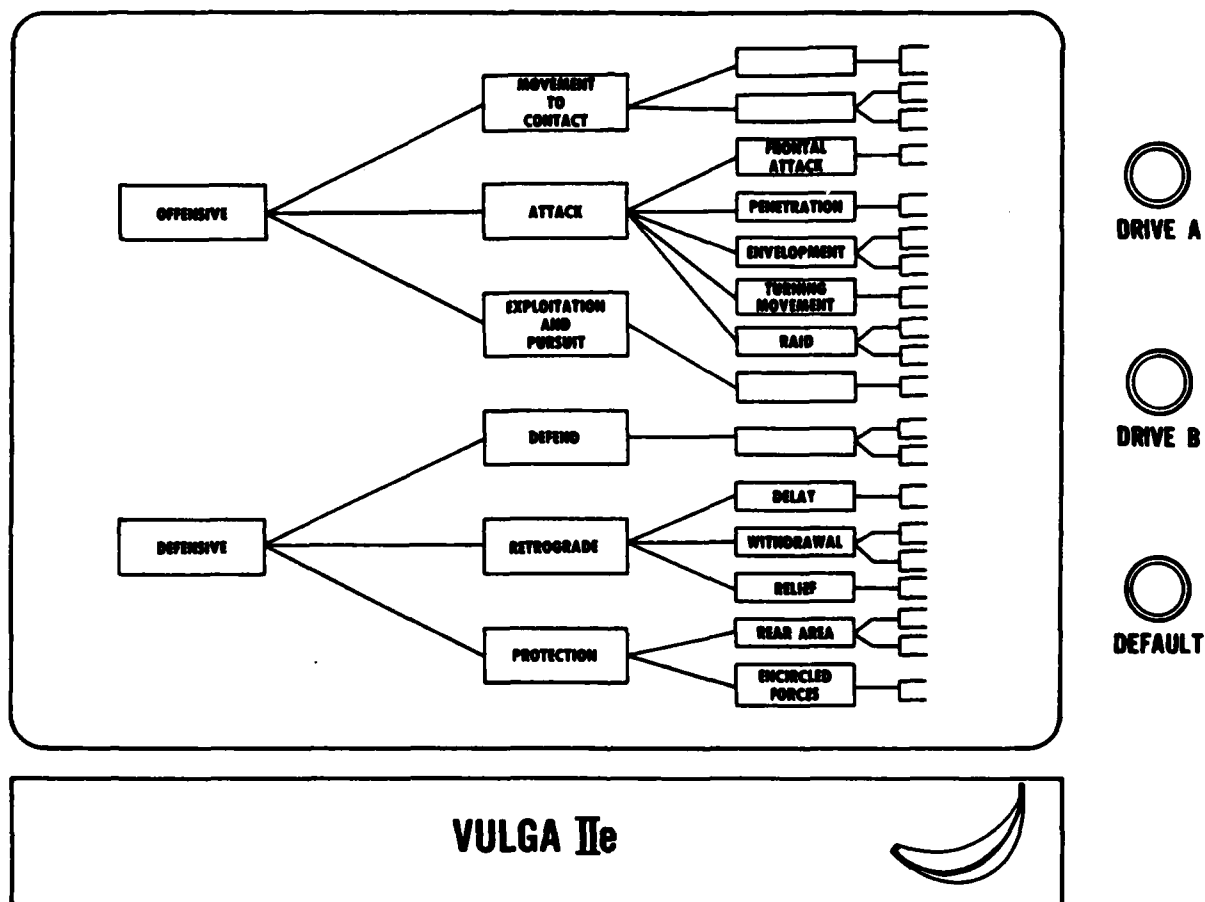


Figure 13

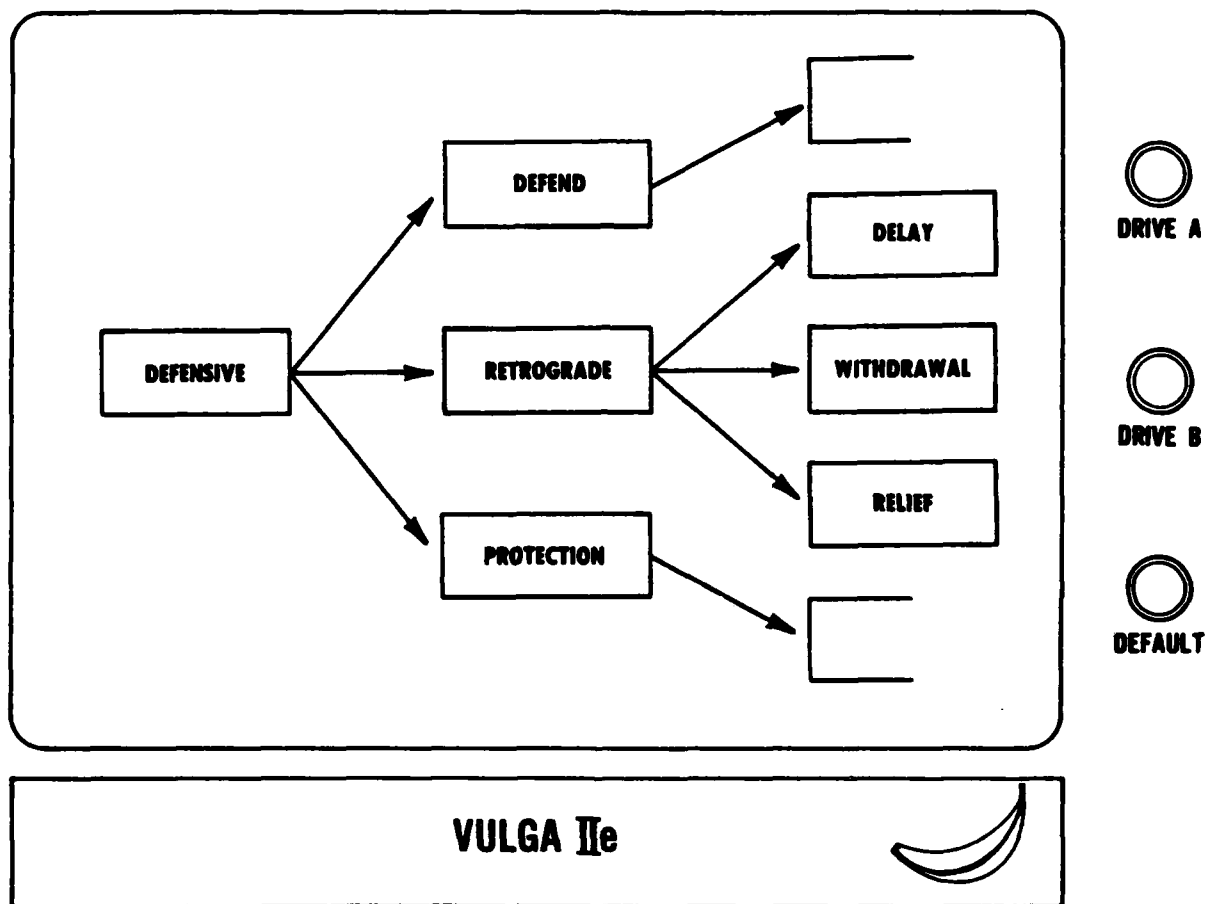


Figure 14

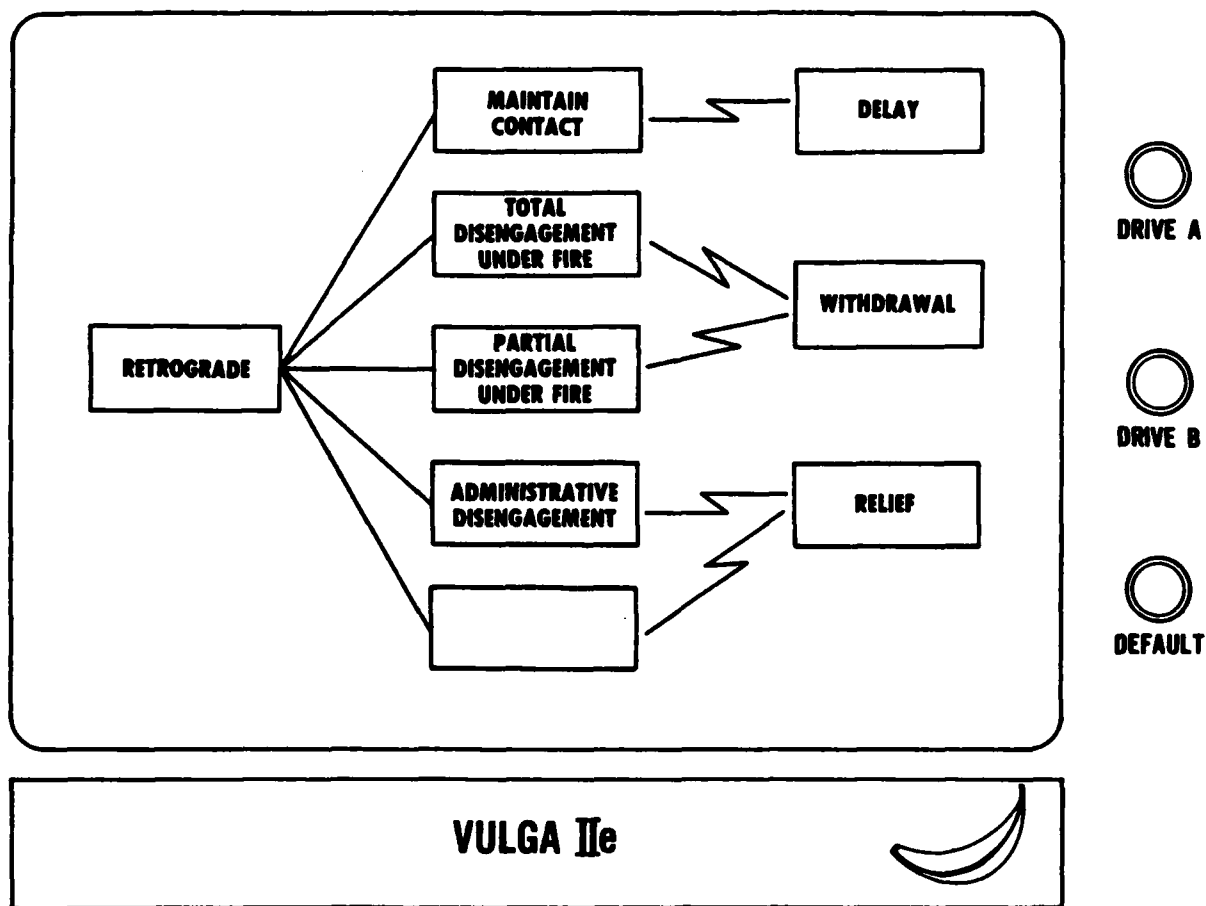


Figure 15

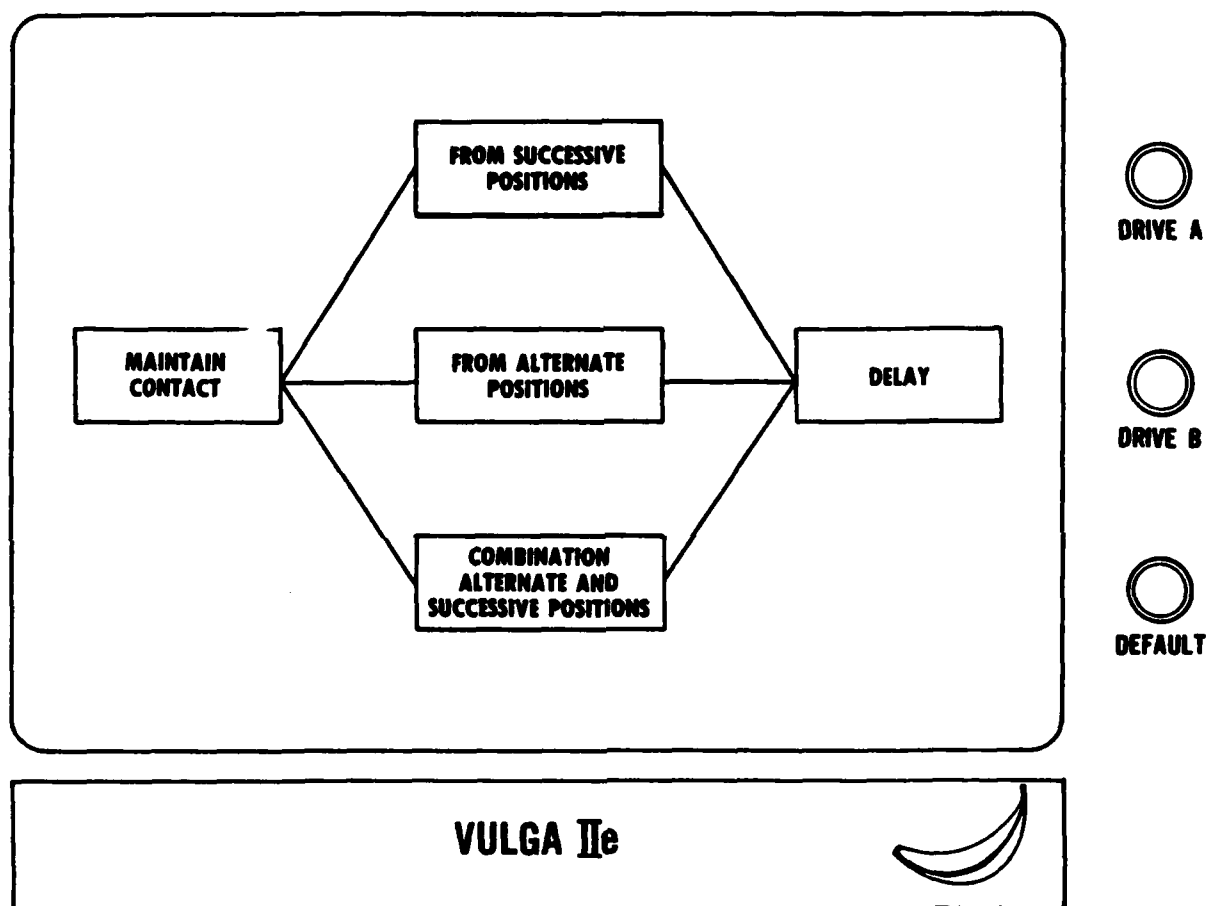


Figure 16

NEGATIVE INDICATORS

DELAY FROM ALTERNATE POSITIONS

1. BARRIERS USED	10. SHORT VHF LINKS
2. FORTIFICATIONS USED	11. SIMPLE MANEUVER CONTROL
3. PRIORITY OF DIVISIONAL FIRES	12. LIMITED RECONNAISSANCE
4. JAMMERS 5+ KM FROM FLOT	13. NO CORPS FO NET
5. ARMOR DUG-IN	14. ADA IN DIRECT SUPPORT
6. ONE MANEUVER UNIT	15. NO CEWI VHF/HF LINK
7. LIMITED MOBILITY	
8. TASK ORGANIZED	
9. EMPHASIS ON LANDLINE	



VULGA IIe



Figure 17

HAS 1/37 INF BEEN USED IN PREVIOUS DELAYS?

1/37 INF USED	90%	7ID DELAY —	UNIT ID
1/37 INF USED	82%	7ID DELAY —	NO ID
1/86 INF USED	12%	7ID DELAY —	UNIT ID
1/86 INF USED	4%	7ID DELAY —	NO ID

ANALYSIS INDICATES 93% PROBABILITY 1/37TH USED IN THIS DELAY

DRIVE A

DRIVE B

DEFAULT

VULGA IIe



Figure 18

GIVEN A CERTAIN AMOUNT OF TIME TO CONSIDER A DECISION, THE MORE TIME SPENT EVALUATING THE DECISION, THE LESS TIME SPENT LOOKING AHEAD.

Figure 19

EXPERT SYSTEMS FOR INTELLIGENCE FUSION

R. Peter Bonasso
The MITRE Corporation

ABSTRACT

This paper describes a project whose objective is to demonstrate the feasibility of utilizing expert system technology to support the military intelligence processes on the modern battlefield. The project consists of a software system that fuses sensor and environment data with a priori knowledge of the enemy and applies expertise concerning critical indicators of enemy intentions to observed enemy behavior. The system is to produce transient situation displays and to satisfy the commander's battlefield information requirements. The work is relevant to the design of the second generation All Source Analysis System (ASAS) for the Army and the Air Force's Enemy Situation/Correlation Element (ENSCE).

The project draws on Artificial Intelligence (AI) techniques that have been used to successfully model human expertise in a variety of areas. A rule-based architecture with highly structure knowledge and data representations is developed. It will automatically correlate and integrate reports from different kinds of intelligence sources, respond to intelligence requests such as the commander's Primary Information Requirements and other Information Requirements (PIR/IR), keep requesting agencies apprised of changes in the perception of the battlefield, and justify its actions and answers.

INTRODUCTION

All-source intelligence fusion is the process of correlating, analyzing, and integrating information originating from the diverse collection resources that support the modern battle force. This effort is in support of the force commander who needs to "see" the battlefield, determine enemy intentions, project the impact of the environment on the battlefield, evaluate the progress of the battle, and support the battle in order to operate effectively.¹

Commanders and intelligence analysis in the field make use of conceptual

problem-solving abilities, whereas computers have generally been used for numeric problem-solving. Artificial Intelligence (AI) is a branch of computer science that has focused on symbolic computation techniques for reasoning in complex problem domains and has, over the past 25 years, given computers the capability to do many kinds of conceptual problem-solving. AI techniques are most appropriate when the data for solving the problem are incomplete, unreliable, or changing with time, when the knowledge about the domain is uncertain, and when the search space of solutions is very large.² In short, AI techniques work best in environments that closely resemble

the information profile of the modern battlefield.

The methodology herein describes a program architecture based on these techniques. The fusion module we have designed embodies PIR/IR decomposition, collection tasking, single source correlation, and fusion capabilities.*

PROBLEM DEFINITION FROM A COMPUTATIONAL PERSPECTIVE

The inputs to the fusion module are intelligence reports and requests. Reports are communications of intelligence-related events from the battlefield environment; requests are intelligence information demands from the force commander and his staff.

The requirements for Intelligence and Electronic Warfare (IEW) based on these inputs characterize a computational problem of considerable complexity. In the following sections the information processing requirements of Intelligence Fusion are drawn out and their implications for the software design described.

Incomplete Information

The reports from which a picture of the battlefield is to be constructed do not reflect the totality of the events taking place in the environment but only a small portion of those events. The fusion module must therefore perform plausible inferences to fill information gaps. However, inference-dependent intelligence may

be less reliable than products based on specific reports and known enemy behavior. Therefore, the base of support for intelligence products which must be explicit both to indicate the reliability of the product and to update the product when missing information becomes available.

Unreliable Information

The reports on which intelligence fusion depends may themselves be unreliable. The fusion process will therefore have to carry out probabilistic reasoning and clustering to fuse reports from different sources. This effort to improve the reliability of intelligence by correlation lies at the heart of this intelligence fusion methodology.

Enemy diversion will result in reports that may be highly reliable in terms of sensor error profiles but which represent an unreliable estimate of the enemy's true intentions. The fusion module must be able to discover that it has been misled and correct its actions in the future. This requires that multiple, possibly contradictory hypotheses be maintained during processing and that the explicit base of support for each hypothesis (i.e., an audit trail) be made available.

Time-Varying Data

The modern battlefield is characterized by very high activity and mobility of the combatants, making situation assessment difficult and tenuous. Estimates of enemy deployment based on intelligence reports often degrade rapidly in reliability as a function of time. This requires the reports to be time-tagged and that some procedure for assessing their deteriorating status be available.

* It should be emphasized that we address collection only insofar as it relates to fusion information needs.

Implicit Structure in Data

Individual reports may vary widely with respect to the scope of their content. For example, a report from Theater-level intelligence may state "Soviet 3rd Army is moving North along the Hunfeld-Bad Hersfeld corridor," while a moving target indicator (MTI) report may indicate "A platoon sized tank unit is moving North along the Meisenbach-Odensachsen corridor." The report structures are the same, but the unit structures being reported are several echelons apart, and there is an order of magnitude difference in the sizes of the geographical areas where they are being placed. This suggests a hierarchical organization of the fusion module to distinguish and relate reports dealing with different aspects of the enemy military organization.

Requests for intelligence will also be time imperative and perishable. Their perishability is similar to report perishability, except that their decay must be monitored. Consider PIR that asks "Will the enemy strike at the south flank within the next hour." If the system has already determined, recently, that the enemy will/will not attack in that time, the request is quickly answered. If not, the fusion module must proceed to determine the answer from analysis. If analysis fails to return a sufficiently reliable answer, the fusion module must task the collection system to provide the needed information. This may require a substantial time investment. The fusion module must be able to cut off further efforts on the PIR and report partial results if collection is taking too long.

THE FUSION METHODOLOGY

The fusion methodology we advocate is essentially a set of knowledge-based

systems operating in concert. Figure 1 illustrates the basic scheme. Requests for intelligence arrive from the force in the form of PIR/IR and are interpreted by the inference engine, which first consults databases and the perceived situation map (Sitmap) to see if it can provide an immediate answer. Failing this, the inference engine begins chaining through its knowledge bases trying to derive the answer to the intelligence request. If this also fails, it turns to a special rule set, the meta-rules, to determine if it should report its partial results or if it should task collectors for the information it needs to make the assessment. Reports are processed by the inference engine which uses report processing rules to update the Sitmap. Auxiliary data bases are required for facts about enemy weapons characteristics, etc. Results of PIR/IR are output to the original requesting module.

Representation of Knowledge

Knowledge in the intelligence domain should be hierarchically structured, be semantically factored into roughly independent sub-domains, and be adaptive and self-referencing. The rule-based methodology for knowledge representation described in this section has the desired features. The technique of representation that we propose is a semantically factored frame-based rule representation with a meta-rule component.

The semantic decomposition of the intelligence fusion problem we will follow appears below.

- Report Processing Rules:
Procedures for inferring enemy battle units from intelligence reports.

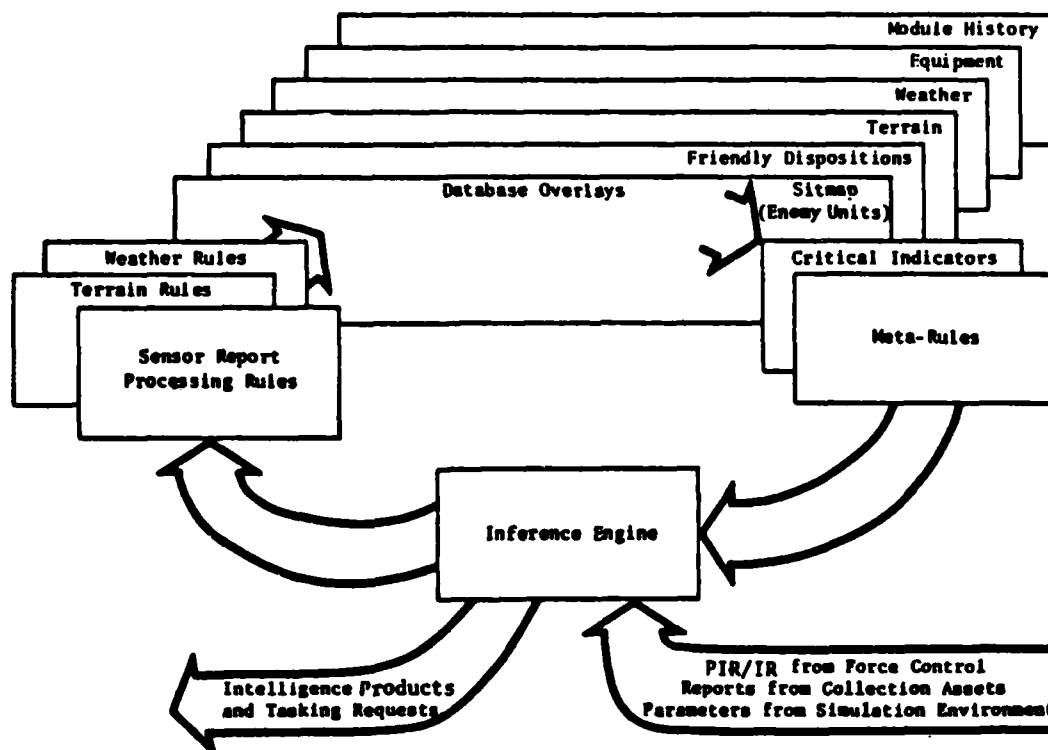


FIGURE 1
A SCHEMATIC OF THE FUSION METHODOLOGY

- **Terrain Rules:** The effects of salient topographical features on model elements.
- **Weather Rules:** The effects of weather on model elements.
- **Enemy Characteristics:** A rule-base of a priori enemy behaviors and critical indicators.
- **Meta-Rules:** A rule set to reason about the system's control behavior, PIR deterioration, tasking, and adaptation.

Report Processing Rules

In recent years MITRE has developed a knowledge-based system for sensor report fusion known as ANALYST.³

ANALYST is a forward-chaining production rule program that processes sensor reports onto a situation map of the battlefield. We incorporate the ANALYST architecture as the report processing component of our fusion methodology—thereby drawing on a well-established, working technology, and simplifying our fusion problem. ANALYST was implemented as a number of small knowledge-based systems operating together. We present the knowledge-bases as subsets of our report processing rule base.

The knowledge base is actually a set of six rule subsets segmented as follows:

- **Cluster Rules** - which gather sensor reports of identical types

and similar locations into activity clusters.

- Pattern Rules - which infer the existence of military units (entities) from patterns of clusters.
- Refinement Rules - which refine unit attributes from tactical or terrain knowledge.
- Merge Rules - which determine when to merge two or more inferred units into a single unit with more refined attributes.
- Reinforcing Rules - which reinforce the inferred existence of enemy units from stray clusters of activity.
- Purge Rules - which purge hypothesized units from the situation map.

A sample from each of the rule sets appears in Table 1.

Terrain Rules

The terrain knowledge-base holds rules pertaining to salient topographical and environmental features and their effect on model elements. We address the terrain representation issue in more detail in the discussion of the databases, below. The terrain representation described there will make possible the efficient application of terrain rules such as :

Terrain Rule 1:

IF the sector is swamp
THEN it cannot traffic wheeled or heavy vehicles.

Weather Rules

The weather knowledge base is similar to the terrain knowledge base in that it holds rules pertaining to the impact of the environment on model elements. An example of a weather rule would be:

Weather Rule 1:

IF it is raining in a sector
THEN MTI reports in the sector are degraded 25 percent.

Enemy Characteristics

In order to successfully analyze and predict enemy activity it is necessary to understand enemy doctrine and order-of-battle. Much of this part of battlefield analysis is embedded in the clustering and pattern rules, but ultimately the statistical and structural aspects of clusters and pattern detection need to be separated and expertise from the respective areas of pattern recognition and enemy order-of-battle and doctrine must be brought to bear. An example of an order-of-battle rule is:

Enemy characteristics rule 1:

IF the unit is a battalion
Command Observation Post (COP)
THEN there are probably two companies approximately 3-5 KM toward the FLOT and one company within a 2 KM radius of the COP.

Additionally there will be a number of sensitive indicators of enemy activity that give early warning of intentions which will amount, operationally, to the specification of a set of expected enemy behaviors. In order to deal

with expected activities, some facility for making tests at future times is needed. An AI technique for achieving this is called "posting demons." The demons are software functions that "wake up," i.e., are called, when an input pattern is matched (such as an

expectation being fulfilled or an alarm going off). A demon causes a special action to be taken. The critical indicators knowledge base will consist of general demons that invoke special actions in special circumstances, and specialized demons that are posted by

<u>Cluster Rule:</u>	<u>IF</u>	the received report is COMINT, and the band is VHF, and there are COMINT clusters within 1 KM of the report with the same frequency
	<u>THEN</u>	add the report to the nearest cluster, and reaverage the cluster's location, and repost the cluster to the situation map at the new location
<u>Pattern Rule:</u>	<u>IF</u>	there exists a COMINT pattern of at least 3 clusters, and each cluster is composed of at least 2 reports, and at least one of the clusters is in the HF band, and the spread between the maximum report count of all the clusters and the average report count is greater than or equal to 3
	<u>THEN</u>	post a tank battalion COP to the situation map at the center of mass of the COMINT pattern and update the entity statistics
<u>Merge Rule:</u>	<u>IF</u>	an entity exists of known type, and another entity exists of unknown type and the sizes of the two are equal and the two entities are within 1 KM of each other,
	<u>THEN</u>	merge the attributes of the second entity with those of the first, and delete both old entities from the situation map, and post the new entity to the situation map
<u>Refinement Rule:</u>	<u>IF</u>	an entity is of type arty, and the FEBA-Distance of the entity is less than 5 KM,
	<u>THEN</u>	change the entity type to unknown, and repost the entity to the situation map
<u>Reinforcing Rule:</u>	<u>IF</u>	the unused cluster is of type ELINE, and its report count is at least 2, and there is an entity with 1 KM whose type is ADA
	<u>THEN</u>	delete the cluster, and update the last-update time of the entity
<u>Purge Rule:</u>	<u>IF</u>	an entity has a last-update time greater than the purge-time, and the entity is stationery, and the entity is not a motor transport company,
	<u>THEN</u>	delete the entity from the situation-map

the consequent part of the rules fired in other knowledge bases. An example of a rule of this type is:

Enemy characteristics rule 2:

IF there is movement of
 additional troops toward
 the front
OR increased traffic toward
 present positions
OR new units have been
 identified in the combat
 zone
OR additional CPs and supply
 and evacuation
 installations have been
 reported
THEN set demon to determine
 enemy artillery disposi-
 tions in two hours.

Meta-Rules

Meta-rules are rules that treat other rules as data. Employing meta-rules can extend the power of a system by giving it a learning capability providing it with an explicit representation for control knowledge, facilitating abstract rule compilation, and making it possible for the system to reason about tasking issues.

It will be necessary for the fusion module to have a limited learning capability for fusion to be correctly modeled. This can be achieved by attaching confidences (a percentage from 0 to 100) to each rule and incorporating rules such as:

Meta-rule 1 (confidence 100):

IF a rule results in a fault
 inference
THEN (recursively) decrement
 the confidence in the rule
 by 25 percent of its contri-
 bution to the inference.

and Meta-rule 2 (confidence 90):

IF the confidence of an infer-
 ence rule falls below 15
 percent
THEN delete and replace it with
 a new rule.

When fired, such rules assign blame to the rules immediately leading to a faulty conclusion, as well as the rules that have led to the firing of those rules. Disfunctional rules are eventually replaced by new hypothesized but untested rules. When such meta-rules are part of a system that processes large amounts of information and that has the means of evaluating its behavior (i.e., adjusting the confidences of its rules) and generating new rules, the systems behavior adapts to its informational environment.⁴

Fact Representation in the Database

In the intelligence fusion domain it is advantageous to use a fact representation that reflects the relationships and structures of the objects of interest - the enemy fighting organization on the battlefield terrain. Frames can provide such a representation. Frames are "object" structures that have characteristics defined by slots. These slots have names and contain values which may be other frames, lists of frames, numbers, etc. The slots in a frame can also be used to specify operations to perform that generate values. This technique is called "procedural attachment" and can be used to represent the effect of the geography and terrain on military units and to do spatial reasoning.

Figure 2 illustrates a possible frame representation for maneuver units. A

request for the strength of the enemy facing the 106th Bn would cause the following sequence of actions. The system fails to find an Enemy-facing slot in the 106BN frame. It therefore looks for an A-Kind-Of slot for an inheritance. This also fails, so it looks for a Superior slot from which to inherit the requested information. This succeeds, but when Enemy-Facing is searched for in the new frame environment it fails again. However, 2nd Armored-Brigade is A-Kind-Of Armored-Unit, and the system tries to derive Enemy-Facing from the Armored-Unit frame. This does not hold the required information either, but Armored-Unit is A-Kind-Of Maneuver-Unit, and Maneuver-Unit has an Enemy-Facing slot with a procedure in it. The procedure is passed down to the 106th-BN and invoked using the FRONT-SECTOR slot local to it. This causes the system to reach out into the correct battlefield sector, and recursively into subsectors, it need be, to estimate the strength of the enemy forces against the 106th BN.*

The integration of the battlefield model is completed by keying the friendly disposition of forces to a hierarchical geographic representation. A frame of the friendly force is created for, say, a Corps. The Corps has responsibility for a front divided

into two sectors, each covered by a division.* Each division front is divided into brigade sectors, those into battalion sectors, and those finally into company-sized sectors of the front. Suppose another "Enemy-facing" request is made, but this time to the 2nd Armored-Brigade. The system accesses its "Enemy-Facing" slot and eventually finds the procedure that adds together the entries in the "Enemy units facing self" slots of its battalions. The system then accesses the battalion "Enemy-Facing" slots and finds it has to determine Enemy-Facing of the subordinate companies. At the company level there exists an attached procedure that accesses the situation map directly to find the known or suspected enemy units near to it. Once found, the results from the company level are passed back up to the battalion level, and from the battalion level to the brigade, where they are combined.

The same structure can also automatically organize enemy activity into meaningful intelligence summaries. This is illustrated in Figure 3, which sketches the front decomposed as described above. An enemy attack occurs as shown. At each level the situation is assessed relative to the force sizes at that level. Thus a single very heavy attack at the lowest echelon may be viewed

* Notice that in climbing the inheritance graph, we needed to know precedence relations among frame slots. Had "superior" taken precedence over "A-Kind-Of", and had "Enemy-Facing 3rd-Division" been previously established as "Soviet 2nd Army", then the incorrect inheritance of "Soviet 2nd Army" for enemy facing 106th BN would have occurred.

* Actually, the Corps is represented as having a perimeter, and its component units as having sub-perimeters, but the "front" representation is a convenient simplification. The Order of Battle representation by unit perimeters rather than by front sectors facilitates the modeling of pockets and islands of force elements required by Air Land 2000.

```

MANEUVER UNIT:
    TYPE: infantry
    SUPERIOR: -
    SUBORDINATES:
    HQ_LOCATION: ((4 Km behind front-sector) (Reliability = .3))
    FRONT_SECTOR: -
    ENEMY_FACING: if (null FRONT_SECTOR)
        then 0
        else if (Null SUBORDINATES)
            then  $\Sigma$ (STRENGTH * ENEMY in FRONT_SECTOR)
            else  $\Sigma$ (ENEMY_FACING * SUBORDINATES)
                + (ENEMY 2nd ECHELON * SELF)
    STRENGTH: if SUBORDINATES
        then  $\Sigma$ (STRENGTH * SUBORDINATES)
        else 1
    WEAPONS: -
    MOBILITY: High
    A_KIND_OF: Combat_Unit

```

FIGURE 2a
MANEUVER UNIT FRAME

as minor enemy activity several echelons up.

The perceived battlefield situation is represented in a system-generated and a system-maintained knowledge base of the current, derived knowledge called the situation map (Sitmap). The Sitmap is a frame with four slots, one for each quadrant of the battlefield. These slots either hold pointers to quadrant frames or are picture element (pixel) arrays representing the pieces of a battle map. The pixel

arrays overlaid with reports clusters that indicate enemy activity (Figure 4).

Our fusion module uses similar data structures to perform terrain and weather analysis in conjunction with the military organization data structure. These constitute a hierarchical set of registered terrain "images" of decreasing resolution matched to the echelons of the battle force. The resolution at each level is dependent on the least area a unit of the corres-

ARMORED_UNIT:

TYPE: armored

WEAPONS: M-60

A_KIND_OF: Maneuver_unit

FIGURE 3b - ARMORED UNIT FRAME

2nd_ARMORED BRIGADE:

SUPERIOR: 3rd_Division

SUBORDINATES: (106th_BN, 107th_BN, 108th_BN)

HQ LOCATION: (6025 7956)

FRONT_SECTOR: ((6031 7953) (6029 7961))

WEAPONS: M-1

MOBILITY: Very High

A_KIND_OF: Armored_unit

FIGURE 3c - 2nd_ARMORED_BDE FRAME

106th_BN:

SUPERIOR: 2nd Armored_Brigade

FRONT_SECTOR: ((6031 7953) (6031 7956))

STRENGTH: 45

**FIGURE 2b
106TH BN FRAME**

ponding echelon would cover. The idea is illustrated in Figure 5. Such structures, derived from the "pyramids" and "quad trees" of the Computer Vision discipline,^{5,6} allow fast local operations to be employed for determining global unit distributions, enemy activities, etc. These data structures will greatly simplify deployment analysis and detail can be carried out in a straightforward

manner. This considerably simplifies the implementation of a rule hierarchy, allowing induction and generalization to be represented as a mere stepping up in the hierarchy. The automatic inferencing and abstracting capabilities of a hierarchy are especially attractive in light of the hierarchic nature of the threat that the fusion module must characterize.

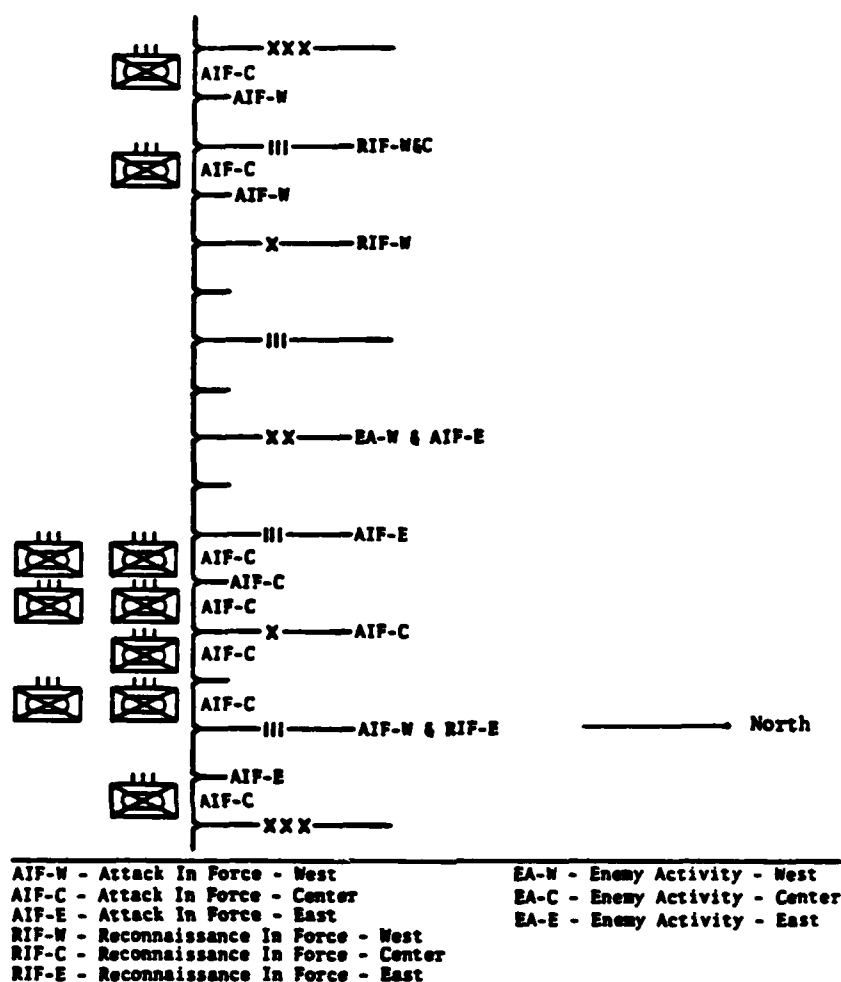


FIGURE 3
THE DECOMPOSITION OF INTELLIGENCE SUMMARIES BY SECTORS

The Inference Engine, Mechanics

The inference engine shown in Figure 6 turns the static structures in the data knowledge bases into result-generating processes. The inference engine is basically a pattern matcher that decomposes the incoming messages into strings that also occur in the fusion module's data or

knowledge bases. The occurrences most often are partial matches. For instance a request may read "How many enemy tank battalions face the 3rd (Brigade)?"

Notice that the inference engine must translate an input from a simulation module into an internal process (e.g.,

Sitmap frame:
 quadrants: (S2 S3)
 reports: (MTI 30 ELINT 20 HUMINT 5)
 activities: (Radar 6 Artillery 10 Maneuver 12)

S2 frame:
 quadrants: (S23 S24)
 reports: (MTI 15 ELINT 10 HUMINT 5)
 activities: (Radar 6 Artillery 7 Maneuver 5)

S23 frame:
 quadrants: (S232)
 reports: (ELINT 7 HUMINT 5)
 activities: (Radar 5 Artillery 5)

S24 frame:
 quadrants: (S243)
 reports: (MTI 15 ELINT 3)
 activities: (Radar 1 Artillery 2 Maneuver 5)

S243 frame:

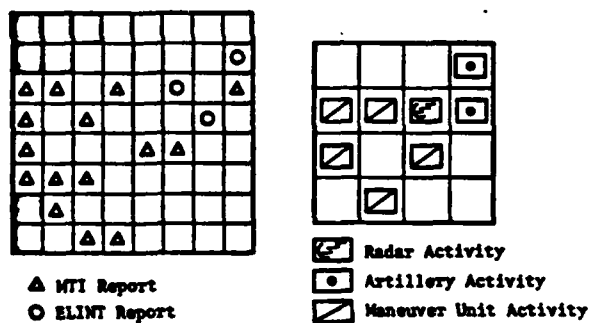


FIGURE 4
 SITMAP REPRESENTATION

"how many" = summing of counts) and also needs to translate a specification derived from another module's format into its own representation (e.g., "facing the 3rd (Brigade)", into the "sector" representation). The inference engine is in respect data-driven. The input determines the procedure invoked. If the message is

an intelligence report, for instance, a database updating operation is required.

The inference engine must know not only how to apply the operations that appear in the rules, but also how to determine the reliability of a statement that is the consequents of other statements of varying probabilities.

The fusion module may not immediately "know" the answer to the request. This will be the case when the request is a "higher level" information that only appears in the rule consequents and (possibly) antecedents. The inference engine then turns to the knowledge base appropriate to the request and matches the transformed request statement to the conclusions or "consequents" of the rules in the knowledge base.

More than one rule may match a request. When this occurs the matching rules are said to conflict. In this case the inference engine needs to be able to reason about the rules in the conflict set to prioritize them. This can be done implicitly with an intrinsic ordering on the entire rule set or by meta-rules which provide ways of explicitly reasoning about conflict resolution. We will follow the latter approach.

The inference engine can use "default reasoning" to make a good guess at what the values might be or it will task the IEW collection modules to provide the missing information. It may also invoke knowledge about the reliability of its default reasoning and the probable cost in time and attrition of the tasking effort in order to decide which action to take.

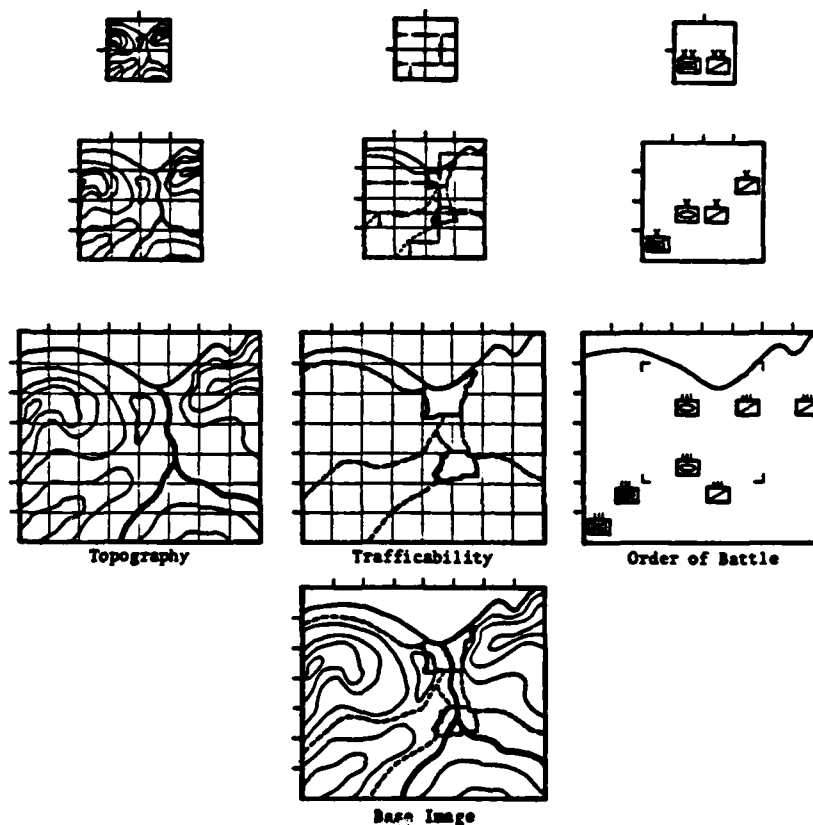


FIGURE 5
OVERLAYED DATA BASE REPRESENTATION

Suppose the request "Can the enemy 103rd Regiment make a deep penetration" is received and it fails to match in the database. The inference turns to the knowledge base and, let's say, finds a rule that states:

IF the unit is well supplied
 and the unit has air support
 and the unit has a decisive
 advantage in armor

THEN the unit can make a deep
 penetration (with
 confidence 70)

Now the inference engine tries to establish the antecedent premises by matching their forms to the database

or knowledge base contents exactly as it did with the original intelligence request. It literally asks itself if it knows or can determine each antecedent in turn. When it tries to determine "the unit has a decisive advantage in armor" it might find another rule that states:

IF (3x the number of armored
 units of size x in the unit)
 is greater than (2x the
 number of armoured units
 of size x facing the unit:

THEN the unit has a decisive
 advantage in armor (with
 confidence 85)

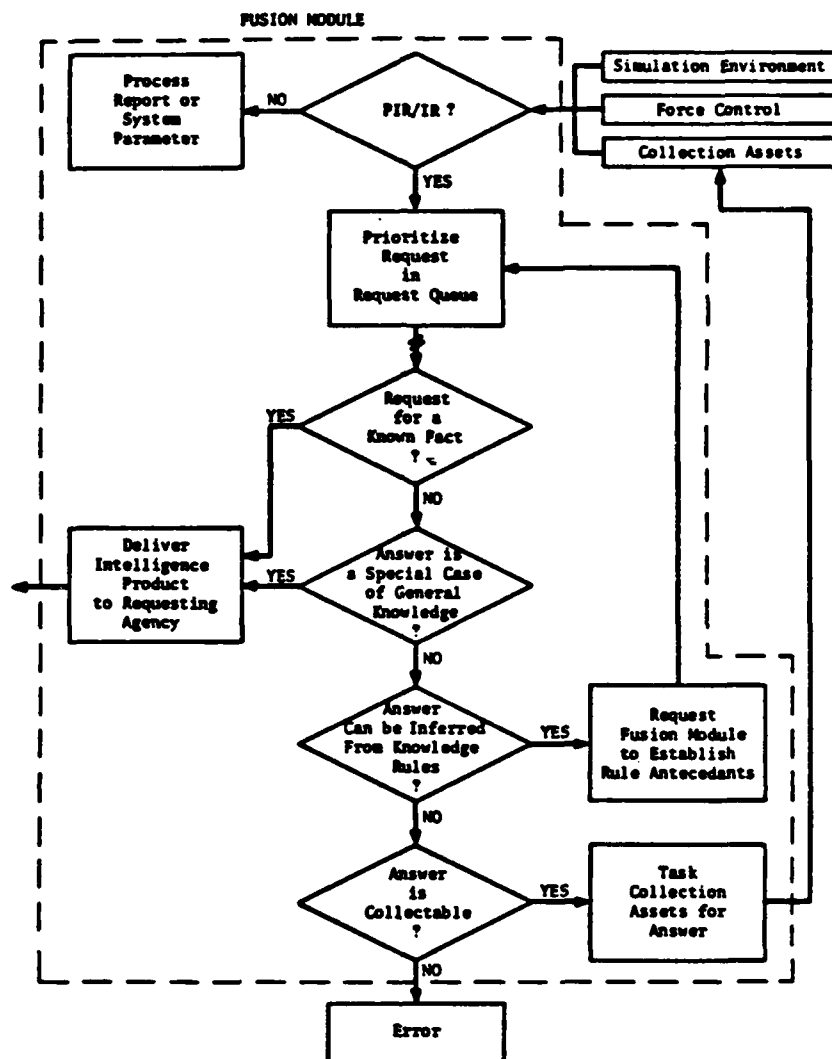


FIGURE 6
INFERENCE ENGINE FLOW OF CONTROL

The inference engine is back to the level of the facts in its database and can proceed to determine the values of those facts or the tasking required, as described above.

HOST SYSTEM REQUIREMENTS

The system described is intended for a mini-computer. Although there is precedent for implementing a capable

production system on a personal computer,⁷ many of the features that make this architecture attractive for analysis may have to be jettisoned due to lack of space.

There is also precedent for implementing systems similar to this fusion system in languages such as PASCAL⁷ and FORTRAN⁸. These languages are extremely cumbersome

for many of the techniques that are appropriate for this task.

The LISP family of languages provide the required symbolic, dynamic, and flexible constructs for implementing the above methodology, and we recommend it for this module. Moreover, small and fast workstation LISP computers are being marketed that will make the choice of LISP practical even for application requiring speed and portability.

CONCLUSION

The methodology presented depends on recognizing two different kinds of demands on the fusion module - integrating intelligence reports and responding to intelligence requests. We employ forward chaining through a rule-base for the former, and goal-directed backchaining for the latter. Both facts and rules are represented in frame structures, making it possible to apply rules to rewrite rules and to explicitly manipulate the control strategy.

The hierarchical frame representation of objects and entities of interest to the intelligence endeavor provides a natural solution to the incomplete data and implicit structure problems. Reliability is explicitly estimated, dynamically adjusted, and accounted for in the continual updating of data items in rules. An evidential reasoning approach is the anticipated solution for the reliability assessment problem. The dynamic aspect of input data is modelled by the dynamic adjustment of reliabilities and the dependency structures of system knowledge that is an integral part of rule-based systems.

The fusion methodology we have presented is an application of well-tested software technology to the difficult problem of intelligence fusion. With direct implementation and proper knowledge engineering this technique will result in an improved model of human intelligence activities on the battlefield. With the features of spatial and temporal data structures, evidential reasoning, and the infusion of intelligence expertise, this technology can result in an extremely versatile system of high capability and utility to the Army community.

REFERENCES

1. Lockheed Missiles and Space Company, Inc., The Determination of Essential Elements of Information for an Army Corps, LMSC-D664802, November 1978.
2. Xerox Palo Alto Research Centers, The Organization of Expert Systems: A Prescriptive Tutorial, VLSI-82-1, M. Stefik et al., January 1982.
3. The MITRE Corporation, ANALYST: An Expert System for Processing Sensor Returns, MTP-83W00002, R. P. Bonasso, 1983.
4. Holland, J. H., Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
5. Antonisse, H. J., "Image Segmentation in Pyramids", Computer Graphics and Image Processing, August 19, 1982.

6. Shneier, M., "Two Hierarchical Linear Feature Representations: Edge Pyramids and Edge Quad-trees", University of Maryland, TR 961, October 1980.
7. Bonasso, R. P., "Running ANALYST on an APPLE", Working Notes, The MITRE Corporation, December 1982.
8. Weiss, S. M., and C. A. Kulikowski, "EXPERT: A System for Developing Consultation Models", Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, 1979.



AN ARCHITECTURE FOR THE APPLICATION OF AI TECHNIQUES TO THREAT WARNING

DEAN F. BABCOCK
SRI International

The techniques developed in artificial intelligence (AI), specifically expert systems or knowledge-based systems, show promise of making a major contribution to survivability and mission success of army vehicles. It is a challenge to bring that promise to fruition as army vehicles go into action against an advanced threat. This paper reflects work performed at SRI International in artificial intelligence, electronic warfare, and multiprocessor computer systems for the Air Force, DARPA, industrial clients, the U.S. Navy, and the Navy of the Federal Republic of Germany, and on institute research and development funds.

The threat systems appearing in our threat studies number in the thousands and are more diverse than the threat systems postulated five years ago. New systems are present with old systems. Some computer-controlled systems have speed and agility that would have been unattainable by our enemies five years ago. We can also expect systems produced by our allies and produced and exported by the U.S. to be turned against us. We need to consider the diversity of guidance present on missile systems designed to work day and night, all weather, and in spite of obscurants.

A major consideration in employing AI techniques is the reduction in crew complement required by the smaller vehicles of the Rapid Deployment Forces or the Division Light forces

(e.g., the one-man-crew attack helicopter). Thus AI systems are required to take over human roles to reduce the size of the crew.

The subject of system progression poses the question: Is there a system architecture that is transferable between users? The two primary users in the Army are aviation and armored. An architecture is required that is transferable between users and between theaters, one that can accept technology insertion, to make use of the doubling of speed and capability of digital equipment that tends to occur about every two years. SRI's study (1) examines the mechanisms to bring artificial intelligence knowledge-based systems and available and postulated resources together into a system concept, and (2) estimates the feasibility and effectiveness of that concept.

The ultimate objective is to develop a system that will perform the role of one or more crew members. The system must have access to all the pertinent information on the mission, the vehicle, the threat, the doctrine, and the available responses. It must provide a competent response to enemy threats in a timely manner. This objective would be attained by an evolutionary process from development to acquisition.

The resources available on board the vehicle to counter the threat are (1) mission data, (2) sensor data, (3) expert knowledge, and (4) responses.

Data available for making employment decisions include (1) mission data (targets, routes, terrain, EOB, friends, foes, neutrals), (2) vehicle state data (position, speed, attitude, weapons, systems, time, fuel, stores), and (3) sensor data (radars-terrain-bomb/Nav-fire control, warning receivers, IR/UV/optical, acoustic, communications). The mission data relate to the objective of the vehicle on that particular flight and are vehicle-dependent. The vehicle-specific data describe the vehicle's present state. The sensor data update the present knowledge of the threat. The knowledge of experts, in the form of rules for system decision making, relates these data to the selection of an appropriate response.

The threat warning system concept can be divided into three levels on each of two axes and categorized in the speed or time domain. The three levels are like the three loops of an autopilot. The countermeasures response is like the inner loop of the autopilot; it is measured in microseconds or seconds. The maneuvers loop is like the navigation loop in the autopilot; its response takes seconds or minutes. The lethal response level is like the fire control loop, which takes seconds to tens of seconds. Dividing the threat warning system this way yields useful insights into the consideration of time/speed demands and sophisticated/smart responses to the challenge.

Mission data essentially consist of the preflight briefing information that would be given to the pilots and crew members, with ground-to-air updates when possible. Vehicle state data consist of the parameters listed above plus others, including weapons and expendables, etc. Sensor data provide information about the environment

and the threat, and consist of active and passive sensor outputs.

Expert knowledge can also be divided into three levels, again according to response time. The first level is instinctive behavior, which is a very quick response, like the automatic response of animals. This self-directing response is unsophisticated, but smart. A useful aspect of machine responses is that they can be re-optimized for the environment more rapidly than humans can.

The next level of expert knowledge has to do with conditioned behavior, where performance results from training in developing skills for competent execution. Analysis is included, but it is coordinated as part of the action and does not affect response time. Most athletic prowess consists of such conditioned response, as do the movements of a skilled pilot. The third level is mission optimization which is measured in a much longer time frame. Here we have thoughtful and considerate action, based on in-depth analysis of background and environment situations, simulation, and reactions. The question that we ask here is "what would happen if?" The answer involves a level of forward analysis or simulation.

The three levels and two axes are shown in Figure 1. The horizontal axis represents three threat types with different response times. The times involved in the terminal threat may be as short as one second. The tactical threat may involve tens of seconds, and the theater threat may involve hours. The vertical scale is the interface axis. This axis has three levels: human interface, analysis, and machine interface.

Interface	Threat		
	Terminal	Tactical	Theater
Human	<u>Inputs</u> Templates Response Doctrine	<u>Inputs</u> Response Patterns Priorities Reactions Doctrine	<u>Inputs</u> Situation Objectives Threat Deployment EOB Doctrine
	<u>Outputs</u> Alarms Visual Aural	<u>Outputs</u> Alternatives Jeopardy Mission Impact	<u>Outputs</u> Routes Tactical Alternatives Advisories
Analysis	<u>Reflex</u> "Knee Jerk" Response	<u>Conditioned Response</u> Considered Response	<u>Thoughtful Action</u> Alternate Plans-- Mission Optimization
Machine	<u>Immediate Automatic Operation</u> < 50 Rules	<u>Analysis Synthesis Simulation</u> < 1000 Rules	<u>Correlation Searches Min / Max</u> ~5000 Rules

FIGURE 1 RELATIONSHIP OF THREAT CATEGORIES TO ANALYSIS LEVELS

In the upper left corner the inputs are templates, response doctrine, etc. The outputs are visual or aural alarms. The analysis level shows the reflex, "knee jerk" response (i.e., stimulus appears--response is taken). The machine interface level shows immediate automatic operation. I have estimated that at that level perhaps 50 rules need to be searched.

In the upper right corner under theater response the human interface can be comprehensive. Technology is available to provide color displays, maps, and graphics for the selection of alternate routes, for making tactical decisions, or giving advisories on targets. The middle level shows thoughtful action with alternate plans and mission optimization. The machine level

shows such things as correlations, searches, min/max solutions, simulations, etc. Possibly as many as 5000 rules are indicated. The arrows in the analysis level imply a flow of information from the reflex level into the conditioned response level. For example, when a reflex action is taken, such as the initiation of jamming, the fact would be known to the conditioned response level and that level would then change its response and start an appropriate action. This could be a maneuver, or a weapon system response. In a similar cross-feed between the center and righthand box, the conditioned response actions may cause a change in the mission optimization.

Knowledge-based systems have evolved over the years to emulate an expert, a plurality of experts, a panel of experts, and the expert team. I have showed this evolution in a circular fashion in Figure 2, with a simple

one-expert system in the upper left corner. Here a single inference engine is working with a single knowledge base and delivering an output. The top center system is composed of two specialists with a controller that feeds information to them according to their roles. They share one knowledge base, each expert specialist coming up with his own decision. On the right, the system has two independent experts, each with his own knowledge base. They are shown consulting with one another, with the arrow joining them indicating consultation. So now both arrows point in the same direction. The lower left corner shows the expert team, consisting of a controller tasking three different specialists, each with his own knowledge base and a shared knowledge base. This is called the clinic system, in that the boundaries between the knowledge bases are soft boundaries, like a group of doctors using the same set of patient records.

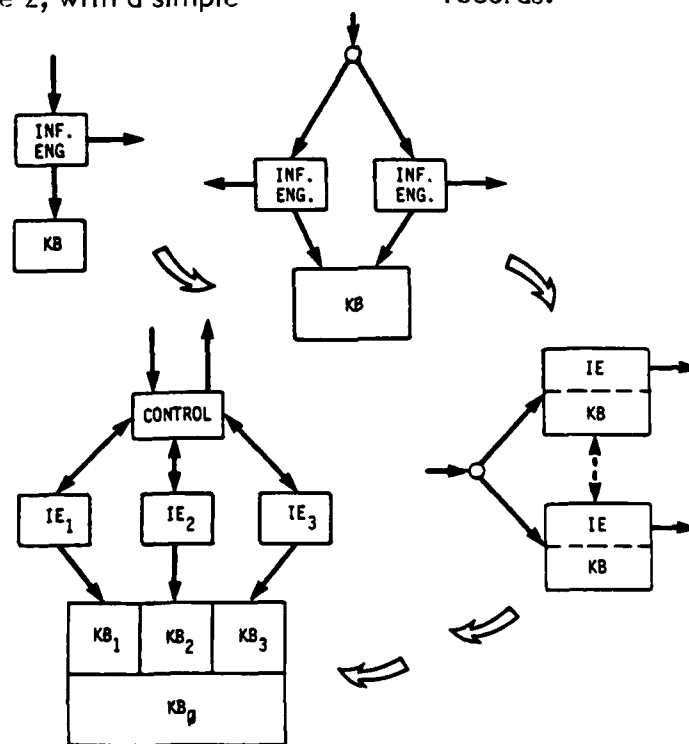


FIGURE 2 CONCEPT DEVELOPMENT

Figure 3 shows one of the knowledge-based system architectures which will be investigated in our defense system study. It consists of four specialists: one each of three category specialists, and a management specialist whose specialty is generalities. The vertical arrows between the inference engines indicate the tasking going up and reporting going down. The vertical arrows in the knowledge base indicate the flow of knowledge from the general knowledge base to the specialist's knowledge bases and back down as the situation changes. It is the function of the management specialist represented by inference engine zero to make the knowledge available to the specialists as the situation changes and as he tasks them. It is also his function to fuse the information and decisions that come from the specialists to reduce contention.

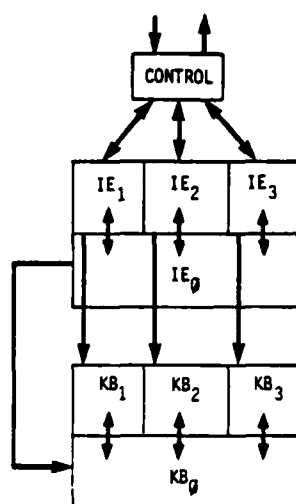
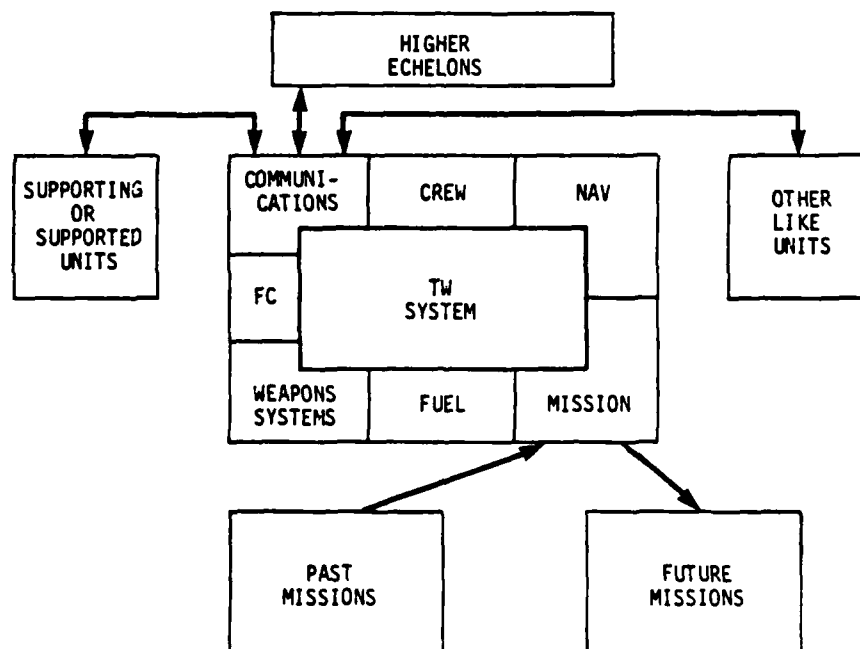
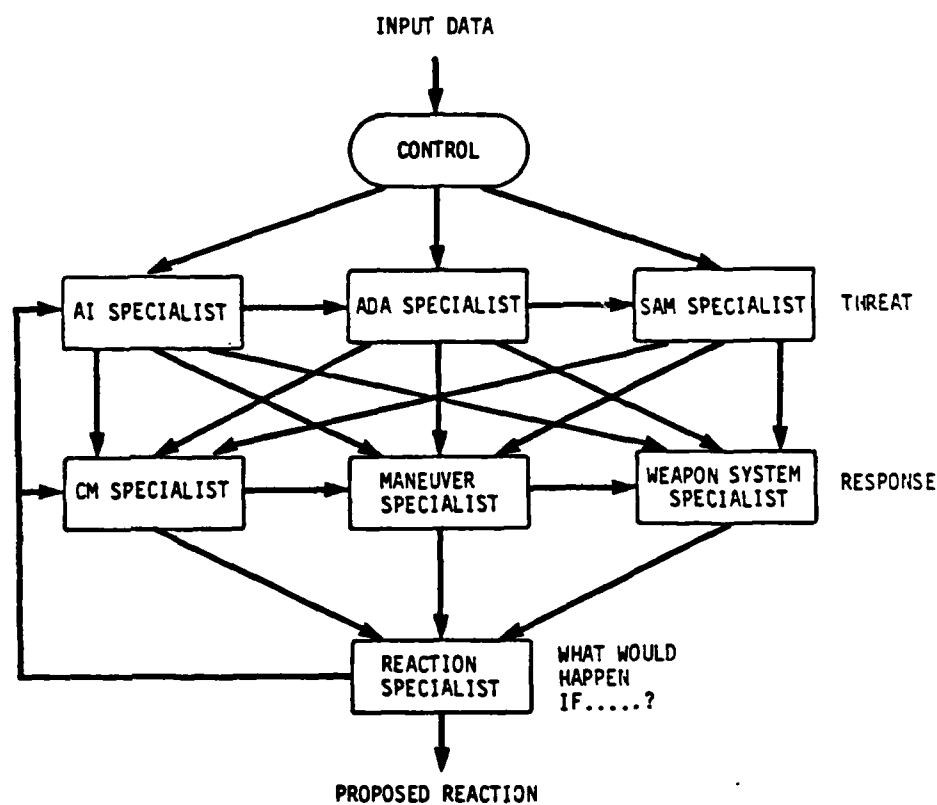


FIGURE 3 KNOWLEDGE-BASED SYSTEM ARCHITECTURE

Figure 4 shows the flow of control in a system which has seven specialists. Envision a system in which there is parallel processing by both threat and response specialists. The controller tasks the information to the three threat specialists: an airborne intercept specialist, an air defense artillery specialist, and a SAM specialist. Their outputs are fed to the three response specialists: a countermeasures specialist, a maneuvers specialist, and a weapon systems specialist. The maneuver specialist is usually referred to in human terms as "the pilot". All of these reactions are reviewed by the reaction specialist, whose job it is to assess the level of jeopardy, to eliminate contention, to look at the question of "what would happen if?" and to give feedback to the threat and response specialists, and select a proposed reaction. The reaction specialist keeps track of a state variable that is known as jeopardy.

In addition to the response decision making, there are also the relationships that such a system needs to maintain. In Figure 5 the threat warning system is shown in the center of the vehicle's system array. The system's position in the hierarchy will depend on the intelligence of other systems. In the Air Force Pave Pillar System, for instance, the data interchange system has a fusion module and might very well occupy this central role. The capability of this system obviously depends on the availability of accurate position data, as well as on adequate communications.

The future of AI threat warning systems in army vehicles depends to a large extent on the feasibility of handling human roles in terms of performance and cost parameters. This paper sets forth several preliminary analyses of such systems and has come



up with some numbers to estimate feasibility. In the first estimate of a simplified version of the defense system, 880 rules were estimated. In a more advanced or comprehensive defense system, 1580 rules were estimated. In estimating memory capacity, we used 100 bytes per rule and with a total system library of 5000 rules this gives a memory requirement of 500 kbytes, modest in any modern system. To estimate speed we used an 8-bit micro with a 10-MHz clock rate, using hardware math. Looking at the inner loop (the reflex response loop) with 25 rules in each specialist category, allowing 5 deep searches, and using hardware representative of current techniques, we came up with time response measured in fractions of a millisecond. As far as size is concerned, using modern space technology, the estimate is 10 pounds and a half cubic foot. In the time frame for which we are talking, we find that a system of this size would be adequate for all but the most caustic threat.

The outlook for an AI threat defense management system depends on a

technical and operational interaction. One major consideration is the operational expert input used in the development program. The identification of the sources of expert knowledge and the development and refinement of a knowledge acquisition system are major parts of the program. In the technical implementation, there are aspects that need research and development, especially the management of concurrent specialist systems and the resolution of contention. A very important consideration is the development of operational acceptance. This development needs to proceed step by step in an evolutionary fashion with the development of the demonstrations, the simulations, and the testing. And then more development is needed, more-simulation, and more testing, with the operational people involved. The motto of "build a little, test a little" needs to be expanded to "build a little, test a little, operate a little" in such a way that we are building faith in the system, so that the operational people will accept the role of the AI system in a life and death situation.

AN AI APPROACH TO MULTISENSOR TARGET IDENTIFICATION

Mr. Roland Payne

Advanced Information and Defense Systems

The use of knowledge-based systems techniques for integrating information from several types of sensors to identify targets of interest is discussed. This class of problems has the following characteristics: uncertainty and ambiguity inherent in the data and current manual analysis rules; diversity in form, level of content, and rate of arrival of the input data; time evolution; imbedding in larger problems of interest; lack of ease in modeling; and manual performance. The knowledge-based systems approach consists of: acquiring and organizing knowledge, selecting a hierarchical hypothesis structure appropriate to the domain, organizing "static" data into data structures easily accessed by reasoning/decision making knowledge

sources, constructing knowledge sources that make specific decisions required to formulate and evaluate an hypothesis, designing an efficient control process, and using scenario situations to refine and adjust the developing knowledge-based system. Man-machine interface issues must also be addressed for both the developer and the domain user. Examples from two domain problems are given: the identification of land combat targets using inputs from four different types of sensors (moving target indicators, imaging sensors, radio emitter locator/identifiers, and radar emitter locator/identifiers), and the identification of air targets using inputs from a digital communications net as well as using all sensors onboard an advanced fighter aircraft.

ATTEMPTS AT APPLYING AI TO SITUATION ANALYSIS

Dr. Carl Verhey
US Army Intelligence School and Center

For the purpose of this presentation, situation analysis is defined as the referencing, analyzing, and evaluation of a currently perceived situation and its projected developments in order to give the commander a recommended course of action for successful accomplishment of mission objectives and goals. An important distinction between this problem set and the problems involved in what is commonly referred to as "indications and warning" is that the probable nonapplicability of standard doctrinal templates and tactical prescriptions after the initiation of middle to high

intensity engagement. Although the application of AI techniques to diagnostic problems (for example, hardware maintenance) now seems achievable, the uncertainties (for example, sensor availability, the difference between the perceived and the actual situation, the status of enemy reconstitution, etc.) involved in the situation analysis problem make closed loop (no human interaction) applications unlikely in the near future. A number of attempts have been made at AI (or AI-like) solutions to the problem and these approaches are described.

AD P003028

AN OVERVIEW OF THE APPLICABILITY AND USE OF ARTIFICIAL INTELLIGENCE TECHNIQUES TO THE PROCESSING OF COMMUNICATIONS AND NONCOMMUNICATIONS SIGNALS

Mr. Douglas Walter J. Chubb
US Army Signals Warfare Laboratory

ABSTRACT

The proliferation of communications and noncommunications signal types throughout the world has created a dense and complex signal environment. Traditional statistical or deterministic signal processing techniques cannot effectively process many of these types of signals. To approach comparable human signal processing performance, artificial intelligence signal processing techniques are being applied within this domain. However, artificial intelligence is not a science but a collection of techniques. Theoretical and practical problems arise when using these techniques. This paper discusses these issues.

INTRODUCTION

The date is May 23, 1942. Commander Joseph John Rochefort and his men of the Combat Intelligence Unit, US Navy, have just succeeded in decoding the date-time format of a Japanese ship-to-shore cipher called JN25. Commander Rochefort's group had been using associated enemy target information and knowledge of the Japanese language and its culture and customs to aid them in decoding the JN25 date-time format. Data used included direction finding information, observed troop movements and associated communications, knowledge of the Japanese ship-to-shore message format before the war and detailed knowledge of the Japanese language and its alphabets. JN25 had been extremely important to Rochefort for some time since

recently intercepted messages seemed to have suggested a massive build-up of enemy troops for use in a possible naval invasion of a Pacific island called Midway. On May 23, 1942, Rochefort's commanding officer, Admiral Nimitz, is given the decoded invasion date of June 4, 1942. Indeed, on that date the Battle of Midway is engaged with the Allied Naval Forces emerging victorious. Later, General George C. Marshall, who viewed the battle from Washington, was to call it, "the closest squeak and narrowest victory" (1). Admiral Nimitz gave Rochefort complete credit for providing the keys that made the American naval victory possible.

A similar group of men was to be found in England: the so-called Ultra Group. The "Battle of the Atlantic" was won largely by their efforts in

breaking the German Enigma Code. Likewise, the Japanese Diplomatic code, the Purple code, was successfully decoded by William Friedman, the chief codebreaker for the US Army Signal Corps.

Today, forty-one years later, most artificial intelligence researchers would probably nod their heads wisely and sagely proclaim that such groups had succeeded in their tasks by successfully using such well-known paradigms as multiple knowledge sources and semantic and syntactic networks, implemented using a Hearsay blackboard architecture complete with competing domain experts! Using the hindsight of forty-one years of history, it now appears that aside from their successes, the significant similarity of these wartime efforts was the similarity of their approach to solving the problems. All of these approaches used associated or domain specific information each of which in turn either tended to support or negate a series of guesses or hypotheses about the information content of the encoded data. That hypothesis which globally produced the most coherent, verifiable information was then considered the correct decoding of the cipher. Today, this process is termed a 'cognitive strategy' since it theorizes about and makes use of a series of loosely defined relationships, all of which amassed together, tend to suggest some underlying structure and associated meaning. It is that structure and the general artificial intelligence techniques used to discover it which will be discussed. Specifically, structures to be considered are those commonly associated with communications and noncommunications type signals.

ARTIFICIAL INTELLIGENCE AND NONARTIFICIAL INTELLIGENCE PROCESSING

Artificial intelligence (AI) signal processing techniques may be more clearly understood by contrasting them with non-AI techniques. Non-AI signal processing approaches, sometimes known as strictly statistical, deterministic, algorithmic or noncognitive, do not employ the 'hypothesize and test' paradigm traditionally thought to be used by humans when processing such signals. These approaches always make use of either a great deal of a priori information and/or simplifying assumptions which are believed to be applicable to either the signal or its domain. A priori information may consist of detailed knowledge of the signal structure (e.g., amplitude, phase) or about the signal's domain (e.g., number and types of other signals present locally, types and amount of noise present). It may be argued that the cognitive process and the deterministic process are essentially equivalent in that both require some a priori information to process the signal correctly. Indeed, as Marvin Minsky (2) notes, this is probably true since well-established and understood AI programs eventually become algorithmic. AI has been and continues to operate at the razor's edge of the unknown. However, the important and significant difference between these signal processing approaches may be seen in their relationship between the data being processed and the signal processing techniques utilized. That is, statistics claims basic knowledge and understanding of the appropriateness of various signal processing techniques a priori; whereas, AI generally makes no such claims. The AI process is about establishing an intelligent mapping or

relationship between data and process, then exploiting this relationship by whatever means are at its disposal. Frequently, this may include the use of a standard deterministic approach.

In an AI process, the establishment or determination of the correct processing steps to be used to achieve some signal processing goal may take the form of a search, if all possible conditions are known a priori; or more importantly, it may take the form of cognitive-like behavior wherein the processing steps themselves are developed and created as a function of data and domain. This criteria then can be used to differentiate an AI signal processing approach from a non-AI approach:

- a) AI approach: a program which creates a sequence of processing steps in order to satisfy some goal.
- b) Non-AI approach: a program which executes a sequence of processing steps all of which are known and implemented a priori.

An example of a type of communications/noncommunications signal which could be readily processed by a non-AI signal classification technique includes any highly structured signal where signal classification is made on the basis of observed signal parameters, $x(i)$, $i=1,2,\dots,n$ such that if the domain being considered consists of k such signals, $[S(1), S(2), \dots, S(k)]$, where $S(j)=(x(1(j)), x(2(j)), \dots, x(n(j)))$, then there exists no $S(g)$ such that $x(i(j))=x(i(g))$ for $i=1,2,\dots,n$. In other words, a nearest neighbor statistical approach to signal classification may be used because of the uniqueness of each set of $x(i)$ within the population of $S(j)$ being considered (Figure 1a).

Examples of communications/noncommunications signals which will not yield to non-AI processing approaches includes signals which are highly unstructured; whose domains are considered very noisy/crowded; where the signal measurement processes are 'fuzzy', highly probabilistic, and errorful. Such signals include most natural languages such as human speech and manually sent Morse code. In both of these examples there is consistent uncertainty in the parameter measurement process, the signal detection process, and the language syntactic/semantic structure (Figure 1b).

ARTIFICIAL INTELLIGENCE SIGNAL PROCESSING TECHNIQUES

Since AI processes are quantifiable in terms of their ability to develop and sequence processing steps while processing communications and noncommunications type signals, one may legitimately ask how is this done. The answer is that the data has some global associative structure which may be quantified and exploited. For example, communications type AI signal processing systems typically exploit inherent language structure such as language syntax and semantics. Another approach is to make use of the fact that communication, in general, is a means whereby parties attempt to express their needs or goals to one another. Models of this process have been constructed in order to aid understanding (3). Noncommunications AI signal processes frequently make use of causal relationships such as time and functional intent. Deductive and inferencing capabilities are usually included in these processes. What is being sought in all these processes is the 'best' functional relation-

Non-AI signal processing:

Signal usually structured or domain well-behaved.
Signal parameters form N-tuple which is not ambiguous.
Simple radar pulse train.

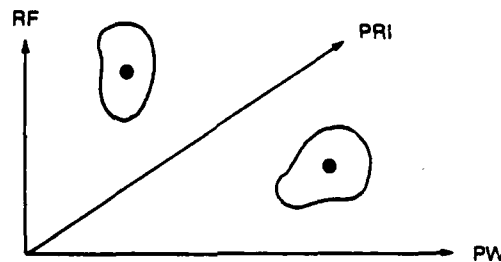


Figure 1a.

AI signal processing:

Signal highly variable, no apparent structure or domain not well behaved. Measurement process "fuzzy" and highly probabilistic. No good error measurement exists. Human speech signal; Manual morse signal.

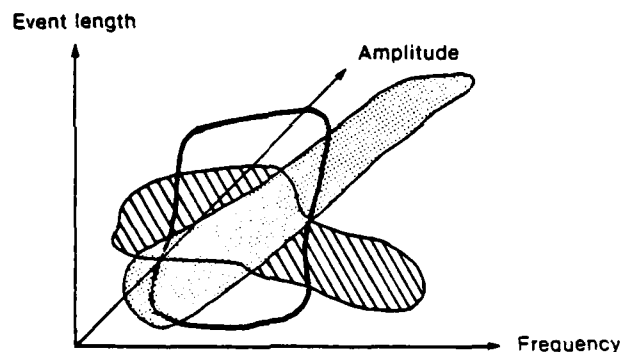


Figure 1b.

ship between data and meaning. This mapping between the so-called 'deep structure' and data continues to be an area of active research within the AI community. This mapping and its characteristics, and its very existence, has been a research topic of unusual concern and activity since this mapping is the structure upon which real intelligence is hung. Numerous related research topics remain to be solved in this regard, such as: 1) What is the local behavior of this mapping? Does it make any sense to ask this question? 2) When is it appropriate to declare some process in error? 3) Is it

possible to make use of this mapping in such a way as to accommodate experience and thereby improve system performance?

ARTIFICIAL INTELLIGENCE PROCESSING OF TACTICAL SIGNALS

These issues are especially significant whenever the signals to be processed are found in domains of crowded,

poorly behaved signals: a military or tactical environment. Such tactical signals may express themselves with erratic signal behavior to include signal fading, sudden and unexpected shifts in frequency, and significant signal structural changes. The domain may contain noise elements which may be momentarily and mistakenly identified as signal. Typical system errors attributable to noise include errors of omission, errors of addition, and system measurement errors. Generally, noise is capable of severely corrupting a signal with the result that the behavior of the mapping between data and the signal's deep structure may be highly erratic and unstable. Fortunately, there are techniques available which can minimize the deleterious effects of noise. Non-AI signal processing systems make frequent use of human assistance or monitoring to reduce the effects of noise, usually at the signal detection stage of system processing. Frequently used human assistance includes various types of receiver adjustment such as bandwidth, signal level and filtering monitoring and assorted types of error processing. In addition, various steady-state assumptions may be assumed about the distribution and types of noise anticipated within the signal domain. Totally automated AI signal processing systems typically deal with the problem of noise by attempting to 'focus' the system on the desired signal, ignoring other signals and noise. System focusing is usually accomplished by constraining the most errorful processing steps with domain knowledge which is believed to be near-certain, a priori. Examples of such domain knowledge include language syntax/semantics for communications type signals and physical law/relationships such as signal reflection, phase, and relational signal parameter behavior for noncommunications type signals.

Topics such as system focusing, learning, and error behavior within severe domains are not well understood by AI researchers. However, to successfully process communications and noncommunications signals which are not exploitable by strictly algorithmic methods; but which are usually processed by human cognitive processes, artificial intelligence techniques offer the best signal processing alternative.

REFERENCES

1. RUSSELL, Francis, "The Secret War", Time-Life Books, Inc., 1981. Time, Incorporated, Alexandria, Virginia
2. MINSKY, Marvin, "Introduction to the COMTEX Microfiche Edition of the Early MIT AI Memos", the AI Magazine, Vol 4 No 1, Winter/Spring 1983, Menlo Park, California
3. COLBY, Kenneth, Mark, "Artificial Paranoia: A Computer Simulation of Paranoid Processes", 1975, Pergamon Press, Inc., Elmsford, New York.

A NATURAL LANGUAGE INTERACTIVE COMPUTER SYSTEM

A. W. Biermann and B. W. Ballard
Department of Computer Science
Duke University

An interactive natural language processor is under development which allows a user to manipulate objectives from any one of several domains on a display screen with typed or spoken English commands. The design of this processor is described with emphasis on features related to noun group resolution.

INTRODUCTION

An interactive natural language processor is under development which allows a user to manipulate objects on a display screen with typed or spoken English commands. This processor is currently being used in a matrix computation domain, where the objects are such things as numbers or matrix rows and columns, and in a text manipulation domain, where the objects are words, sentences, paragraphs, and so forth. After each command, the system responds immediately, usually in one to four seconds, showing the appropriate actions on the computer display. The user can then verify correct action and, if an undesired behavior is observed, issue an "undo" command and then rephrase the original request.

Voice recognition is being done with user-dependent discrete or slow connected speech recognizers. Such systems require the user to provide one or more samples of each spoken word in the vocabulary, and identification of unknown words is done by making comparisons with the pretrained samples. Sentence parsing

is done with ATN style parser and semantics processing employs a procedural representation of world knowledge. A touch sensitive feature has been included on the display screen so that users may augment English commands with graphic inputs.

Such a system allows, for example, a user in an office automation environment to sit at a display terminal and issue such commands as

"Display file alpha."

"Merge this paragraph with that paragraph." (with two touch inputs)

"Capitalize the first letter of each word in the title."
and so forth.

The natural language processor comprises four modules for token acquisition, syntax analysis, noun group resolution, and imperative verb execution. For typed input, the **token acquisition** module scans the input line for words, looks them up in a dictionary, and passes them with their meanings on to the next phase of processing. For voice input, this module interfaces with voice equipment, receives for each input word a set of guesses as to its

identity, and again passes definitions on to the parser.

The **syntax analysis** module receives the identified tokens from the scanner. In the case of voice inputs, the incoming tokens may not be completely identified, in which case a set of guesses is given instead. The parser selects which guesses are most likely to be correct, constructs a sentence from them, and marks all sentence constituents to indicate their relationships to one another.

The **noun group resolution** module finds the addresses of the items being referenced. Most of this work is done by a pair of routines that (a) generate selected objects of the domain at hand and (b) apply any of several modifier types to a previously constructed set of objects. These two routines operate in conjunction with a global mechanism that maintains a "focus list" of objects that have been mentioned recently by the user, thus allowing for incomplete or pronominal inputs.

The **imperative verb processor** constructs the code necessary to complete the requested task and sends this code to the host machine. Each of the above mentioned four modules will be described in more detail in the following sections. For illustration, we will trace the representative input "capitalize the first letter in each word" through each phase of processing.

TOKEN ACQUISITION

For typed input, the token acquisition module scans the input line for words, does some local processing on those

words, looks them up in the dictionary, and passes the results on to the parser. Some local processing is necessary because not all expected vocabulary will be in the dictionary. For example, there are an infinite number of ordinals: first, second, third, ... Local processing would convert, for example, the ordinal "twenty-second" to some standard form such as (ORD 22) and pass that to the parser. Local processing may also be necessary for handling plural forms, for various verb forms, and to provide spelling correction. For voice input, the system receives a sequence of word slots, each composed of a set of words, without knowing the actual user utterance. For example, it might receive the following:

Word	First guess	Second guess
1	capital	capitalize
2	iced	the
3	first	for
4	letter	better
5	nine	in
6	eight	each
7	word	third

In other words, the recognizer has proposed that the first word is either "capital" or "capitalize", the second word is either "iced" or "the", and so forth. In this case, the processor would send all alternatives to the parser, where the selections of those words would be made.

SYNTAX ANALYSIS

The role of the syntax analysis module is to determine the structure of the typed or spoken command. Input consists of a list of "tokens" analogous

to those of a compiler-type scanner. However, as we observed earlier, multiple word meanings will frequently be proposed by the voice recognizer.

Associated with each token recognized is a list of feature-value pairs. Each set of features will include "quote", to allow checking for a particular word; "part", to designate part-of-speech; and additional features, depending upon the token in question. For instance, if the spoken word "two" occurs as the fourth token of a command, the scanner output might contain the following information concerning it:

- 4a: (quote to) (part prep)
- 4b: (quote to) (part casemarker)
(verbcase put)
- 4c: (quote two) (part num) (value 2)

This is the scanner's way of telling the syntax routine that the fourth word of the input is either (a) the word "to" used as a preposition; (b) the word "to" used to signal an operand (case) of the imperative verb "put"; or (c) the word "two" used as a number.

Output from syntax processing will consist of a treelike structure representing the "parse" of the input. Many grammatical categories, such as imperative clauses and noun phrases, permit a slotted, template structure which is compact to store and efficient to access. Thus, for the sample input being considered, syntax analysis will result in a structure suggested by the followings:

Command:	capitalize (imperative)
Noun Phrase:	the (article)
	first (ordinal)
	letter (head noun)
	in * (prep phrase)
	*: each (quantifier) word
	(head noun)

We see here that "capitalize" has been identified as the main imperative verb of the command. Its one operand has been parsed as a noun phrase centered around "letter" with two modifiers: one preceding the noun (the ordinal "first") and one following it (the phrase "in each word"). Other commands having the same superficial form as the one (i.e., Verb, Nounphrase, Preposition, Nounphrase) may involve attaching the word "in" to the imperative verb, rather than to a previous noun, e.g.

"Put the Jenkins file in my home directory."

The grammar used to describe the inputs to be recognized by the system is represented by transition networks similar to the Augmented Transition Networks (ATN's) of Woods. Basically, ATN's generalize the notion of a finite-state recognizer by permitting (a) names of subnets, as well as terminal symbols, as arch labels, and (b) arbitrary structure-building actions. Parsing of an input involves executing the transition networks interpretively. The structures built are used both for constructing the parse of the input being recognized and for saving information about earlier parts of the sentence for later processing.

NOUN GROUP RESOLUTION

Once the constituents of the command have been identified, processing of their meanings can proceed. First, each noun group must be resolved to determine what objects are being referred to and then the imperative verb module can be invoked to operate on them.

Noun group processing begins with the head noun. A meaning representation must be designed for such words as printer, paper, paragraph, sentence, and word. It turns out that the meaning of such words is dependent upon the context in which they are found. If the word "printers" is mentioned by an individual standing in a computer room, it probably applies to the printers in that room. If a salesman mentions the word in a conversation, he may be referring to all the printers, or all the types of printers, manufactured by his company. If one is discussing a particular computer, the word "printer" may designate a particular piece of equipment connected to that machine.

Thus it is necessary to have a representation of the things being discussed. We call this a **focus list**. Conceptually, it is a list of the exact objects being discussed. That is, it contains the physical addresses of the particular items "in focus". At the beginning of a conversation, the focus list will contain only one thing, "world":

Focus: (WORLD).

This "world" will contain many things such as files, calendars, messages, and so forth. If a person would say

"Display my files."

the focus list would then contain both "world" and a list of all the files:

Focus: (WORLD, F1, F2, ..., Fn)

The person might then say

"Display that paper."
"Center the title."

at which point the focus list would include a specific pointer to the designated paper and a pointer to the title.

Focus: (WORLD, F1, F2, ..., Fn, P17, T17)

The focus list is updated each time a noun group is processed and each time a graphic input is received. An item is deleted from the focus list (or given a lower focus priority) whenever there is a new reference to an object of similar type or a containing object. Thus if a person would mention another paper that is not currently displayed, the current paper and all of its contained objects (title, sections, paragraphs, etc.) would be moved out of focus.

Focus: (WORLD, F1, F2, ..., Fn, P6)

Once the focus mechanism is established, one can return to the issue of what is meant by a given noun. Our assertion is that in most cases **the meaning of a noun is the set of objects of the type referenced by that noun in the smallest containing object on the focus list**. Thus if one says

"Display that paper."
"Capitalize the words."

then "words" probably refers to all of the words in the paper. That is, "words" references a type of object in the world and "that paper", P17 for example, is capable of containing that type of object. Thus in this case, "words" would reference all the objects "word" in P17. If one says

"Display that paper."
"Center the title."
"Capitalize the words."

then "words" probably refers to all the words in the title. Here "title" T17 will also be on the focus list, it will also be capable of containing objects of type "word", and it will be smaller than paper P17. There are exceptions to the rule given above for determining the meaning of a noun, but in most cases our processor will use exactly that meaning.

We will use some notation to represent the concepts described above. We define

containerof (X)

where X is a type of object, to be the smallest object on the focus list capable of containing objects of type X. Thus in the above example

containerof (word) = P17

and

containerof (word) = T17

depending on the sentences in context.

We will also need a function to compute the set of objects of a given type. We define

typegen (X,Y,Z)

where X is a type,
 Y is a binary relationship,
 and
 Z is an object,

to be the set of objects of type X with relationship Y to object Z. Thus

typegen (word, in, T17)

is the set of objects of type "word" with relationship "in" to object T17. Let us assume that T17 is the title "a natural language processor" which appears on line 9, characters 20 to 47.

Then typegen (word, in, T17) would generate the set of all words in T17 in an internal form given approximately as

{ (WORD, 9, 20-20),
 (WORD, 9, 22-28),
 (WORD, 9, 30-37),
 (WORD, 9, 39-47) }

We refer to such data representations as **datareps**. In this case, four words on line 9 on a page of text have been selected. The first word has only one letter, the 20th character. The second word is composed of the 22nd to 28th characters, and so forth.

Using this notation, we can now restate the definition given above for the meaning of a noun: The meaning of a noun of type X is equal to

typegen (X, in, containerof (X)).

Of course, noun groups contain many constructions besides simply the head noun. We process many of these with a function called **apply**.

apply (X,Y,Z)

where X is a constituent name,
 Y is an instance of the
 constituent, and
 Z is a set of objects,

is defined to yield the set Z after constituent Y of class X is applied. For example

apply (adjective, animate, { box,
 Jill, Fido })
= { Jill, Fido }

apply (ordinal, first, { box, Jill,
 Fido })
= { box }

```

apply (ordnum, first-two, { box,
  Jill, Fido})
= { box, Jill}

```

```

apply (relclause, RI, { box, Jill,
  Fido})
= { box}

```

if RI is the relative clause "which contains computer manuals" and if the designated box in fact does contain computer manuals. Apply is thus primarily a subsetting function with the subsets being specified by modifiers to the head noun.

One additional function is needed before the meaning of complete noun groups can be represented. We define

```

collect (X)

```

where X is an object to be a function which is initialized to the null set and then called repeatedly with objects X. If a particular call to collect (X) is executed repeatedly, it will build the list of all objects X that have appeared as arguments since the initialization. Thus, if we initialize collect (X) and then execute that call to collect with X=OBJ1 and X=OBJ2, the result will be the list OBJ1, OBJ2. If several distinct calls to collect are simultaneously included in a routine, each call has its own accumulated list.

Using these functions, it is easy to represent the meanings for a large class of noun groups. Thus

"the long words"

is represented by

```

for X in apply (article, the,
  apply (adjective, long,
    typegen (word, in,
      containerof (word))))
collect (x)

```

This program finds the container of words which is in focus, creates the set of words in that object, removes those words that are not long (by some definition), completes article processing which checks certain semantic compatibilities and formats the set, and collects the set of all such objects.

Similarly many other noun groups can be done. For instance, "each word" becomes

```

for X in apply (quant, each,
  typegen (word, in,
    containerof (word)))
collect (X)

```

Certain types of postnominal qualifiers result in nested loop constructions. Thus

"the first letter in each word"

is represented as

```

for X in apply (quant, each,
  typegen (word, in,
    containerof (word)))
for Y in apply (article, the,
  apply (ordinal, first,
    typegen (letter, in, X)))
collect (Y)

```

Hopefully, the definitions of the notations have been complete enough so that the execution of this program is clear. Carrying on the example begun above, where title T17 has just been mentioned by the user, the outer level processing would construct the set of words ((WORD, 9, 20-20), ..., (WORD, 9, 39-47) . Then with X=(WORD, 9, 20-20), the inner loop would be entered finding the set of all letters in X:

```

{(LETTER, 9, 20)}

```

And the first item in the set would be selected and passed on to collect. Additional passes around the outer loop would result in this final collected set.

{(LETTER, 9, 20), (LETTER, 9, 22),
(LETTER, 9, 30), (LETTER, 9, 39) }

Thus we have shown the complete resolution of the noun group

"the first letter in each word"

down to the addressing of the items being referenced.

Indefinite noun groups pose a special set of problems. If a person says

"Output the letter on a printer."

it would appear that there is no concern as to which printer. It would seem that the indefinite article "a" means "pick one randomly". This meaning can be implemented but there is a hazard which we have investigated at length.

Specifically, when a random choice is made, there may be undesirable consequences. For example, once the printer is selected and the imperative verb is invoked to output the letter, it may turn out that the printer is busy or inoperative. The user would be extremely displeased if the command failed while there are available printers on hand.

This is, in fact, a very common occurrence in natural language processors. There are many places where random choices might be made and there are many routines that could fail in processing for one reason or another. A good solution, which has not yet been implemented by us, is to handle all semantic processing with a nondeterministic mechanism that can

select random choices at any point and fail at a later point to back up and try other choices. In the case of the randomly chosen printer, the failure of the output routine would cause a different selection of "a printer" and a repeated attempt to output the letter. All possible choices would be attempted before an error message would be returned to the user.

Another example of the need for nondeterminism is in the sentence

"Put several figures on page 6 or 7."

There are two points for possible nondeterminism here, the definition of "several" and the word "or". Suppose an applications programmer has defined "several" to mean 4 or, if that fails, 3. Then the nondeterministic processor would attempt to put four figures on page 6. If that fails, it would try page 7. If that fails, it would try putting three figures on page 6 and then 7.

IMPERATIVE VERB PROCESSING

Suppose the user has said

"Capitalize the first letter in each word."

We have seen in the previous section how processing of the noun group will yield the set of addresses of objects to be affected.

$S = \{ (LETTER, 9, 20), (LETTER, 9, 22), (LETTER, 9, 30), (LETTER, 9, 39) \}$

The task of the imperative verb processor is to execute the appropriate operation on each object of the operand.

While processing could be quite complicated on some verbs, in this example we would execute

For X in S, capitalize X.

The affected line 9 would thus be changed from

a natural language processor
to
A Natural Language Processor
satisfying the original request.

Adverbial modifiers to an imperative may result in various actions. Thus

"Print the letter twice."

causes an extra loop to be constructed around the print routine. However,

"Print the letter in boldface type."

requires that "print" be executed with a flag sent to the device indicating the special type.

BACKGROUND

Our first natural language processor called NLC was designed for matrix calculation and is described in Biermann and Ballard (80). This was a typed input system which was tested extensively in experiments described in Biermann, Ballard, and Sigmon (83). Voice processing was added to this system in 1982 as explained in Biermann et al. (83) and extensive human factors studies have been

carried out during 1983. Measurements related to rate of speech, error characteristics, success in problem solving, and user satisfaction have been made in an attempt to judge the desirability of such a system in voice interactive problem solving. A report giving results will be released in 1984.

A second effort began in 1980 to construct a generalized voice interactive processor for office automation applications as described in Biermann (82). This system is becoming operative this year as a text manipulation system and will also be instantiated as a file management system. It is voice interactive and utilizes a large screen color graphics terminal with a touch input feature.

A third effort began in 1981 to allow actual users to adapt an existing natural language processor to deal with new data domains. Details on its construction and projected capabilities appear in Ballard (82) and Ballard and Lusth (83).

ACKNOWLEDGEMENT

This work was supported by National Science Foundation Grant MCS7904120 and MCS8113491 and by the IBM Corporation under GSD agreement no. 260880.

REFERENCES

1. Ballard, B., "A Domain Class Approach to Transportable Natural Language Processing", **Cognition and Brain Theory**, Vol. 5, No. 3, pp. 269-287, 1982.
2. Ballard, B., and Lusth, J., "An English-Language Processor that Learns About New Domains", *Proceedings of the National Computer Conference, Anaheim, May 1983*, pp. 39-46.
3. Biermann, A. W., and Ballard, B. W., "Towards Natural Language Computation", **American Journal of Computational Linguistics**, Vol. 6, No. 2, 1980, pp. 71-86.
4. Biermann, A. W., Ballard, B. W., and Sigmon, A. H., "An Experimental Study of Natural Language Programming", **International Journal of Man-Machine Studies**, Vol. 18, No. 1, 1983, pp. 71-87.
5. Biermann, A. W., Rodman, R., Ballard, B., Betancourt, T., Bilbro, G., Deas, H., Fineman, L., Fink, P., Gilbert, K., Gregory, D., and Heidlage, F., "Interactive Natural Language Problem Solving: A Pragmatic Approach", *Proceedings of Conference on Applied Natural Language Processing, Santa Monica, February 1983*, pp. 180-191.
6. Biermann, A. W., "A Natural Language Processor for Office Automation", *Proceedings of the 1982 Office Automation Conference, San Francisco, April 1982*.

EXPERT SYSTEM FOR TACTICAL INDICATIONS AND WARNING (I&W) ANALYSIS

Garo Kiremidjian, Albert Clarkson
ESL/TRW

Douglas Lenat
Stanford University

Adapted from ESL-Q4028

ABSTRACT

This paper addresses the problem of developing an expert system to aid the G2 intelligence analyst in performing the tactical indications of warning task. These include assimilating numerous reports based on sensor and intelligence data (in signal and symbol form); combining such knowledge with information pertaining to terrain, weather patterns, enemy organization, equipment and general capabilities; and predicting significant events in a battlefield environment such as numerical changes of enemy forces, enemy attack at key positions, unusual deployment of forces and equipment, and so forth. The specific problem areas investigated for expert systems technology applications include activation of tactical indicators; indicator relationships (how the activity of some indicators can be predicted/inferred from others); identification of information gaps and collection tasking (identifying the critical elements of information that are not available for making reliable I&W assessments, such as definitive indicator activation and how to task sensors to obtain them); and tactical warning. The approach suggested by the current work consists of incorporating expert knowledge into condition/action rules and developing appropriate frame structures for various categories of sensor and intelligence reports. A global two-dimensional data structure is used to accommodate specific reports from very different sources, efficiently trigger relevant rules, and keep the human analyst abreast of the developing threat situation. Each rule is also represented as a frame, facilitating browsing through the rules, adding new rules, and assigning credit and blame to rules. Such representation is currently a very useful feature in the development of a strategic I&W prototype expert system.

INTRODUCTION

The application of expert systems techniques to the problem of tactical battlefield indications and warning (I and W) appears extremely promising. Discussion with U.S. Army commanders, line officers, and retired

officers has led to emphasis on the following requirements in developing such applications:

- Development of battlefield threat scenarios which distribute tactical indicators chronologically.

- Identification of the types and sources of intelligence reports associated with the various classes of tactical indicators.
- Acquisition of expert knowledge on tactical I and W decision and assessment criteria for the following:
 - a. Indicator activation
 - b. Indicator relationships across time
 - c. Identification of incomplete and/or unreliable report information
 - d. Collection system tasking
 - e. Tactical warning.
- Design and development of a prototype tactical I and W expert system.

BACKGROUND

Based on several years of contract work in the areas of (1) mission planning and management, and (2) strategic I and W analysis, ESL has gained substantial experience in understanding intelligence problems and developing computer-based support for intelligence analysis tasks. The relevant work includes the following:

- a. Several years of work in I and W analysis under contract to DARPA. Work has involved extended on-site research at CINCPAC/IPAC and has lead to an experimental computer-based analytical modeling and management technology.

- b. Ongoing ID projects to develop expert systems for strategic I and W analysis and interpretation of tactical battlefield communications intelligence data for the purpose of communications net reconstruction.

PROBLEM DISCUSSION

Accurate and reliable tactical knowledge about the enemy and the area of operations is vital to command decision making in a battlefield environment. Such information is the product of tactical I and W tasks performed by experienced G2 analysts. They must assimilate numerous reports based on sensor and intelligence data, and combine the reports with facts pertaining to terrain, weather patterns, enemy organization, equipment, and general combat capabilities. Significant changes or events can then be predicted. Important questions include whether the enemy will attack, defend, reinforce or withdraw, and, if so, when, where and in what strength these conditions will occur. It is also important to distinguish enemy intentions and capabilities. Do the capabilities support the indicators or is the enemy trying to conceal his true intentions?

These assessments must not only be dependable but also timely.

Thus, tactical I and W analysis has the following characteristics:

- The problem domain is complex. A diverse set of decision criteria and heuristics is required to integrate the report data from various sensors, factor in

knowledge about the enemy's tactics and modus operandi, and then assess developing threat situations accordingly.

- Experience is essential. The effective performance of I and W analytic tasks requires experience, training, and a detailed data base. The voluminous intelligence data produced by modern collection systems must be meaningfully interpreted and properly assessed in terms of rapidly changing threat environments. This is accomplished primarily by relying on the skills and knowledge of experienced trained analysts.
- Knowledge is dynamic. Assessment of enemy capabilities is based not only on past enemy actions, but also on current enemy force characteristics and the present situation. These elements are influenced by factors such as new weapon systems (which could be in various developmental stages such as R&D, production, development, and so forth), new collection systems, current and developing political and economic conditions, and so on. Such information greatly impacts the need for frequent updates and reevaluation of the knowledge base required for the I and W analytic tasks.

These and other complexities of tactical warning mandate the application of expert systems technology. Expert systems will increase productivity and efficiency in performing timely I and W analytic tasks. Expert systems will also improve I and W training in order to overcome the scarcity of I and W

expertise and the high turnover rate of I and W analysts. The following sections describe some of the necessary steps for such applications.

INDICATORS AND THREAT SCENARIOS

The I and W analyst does not draw high-level conclusions about enemy intentions directly from incoming sensor and intelligence reports, any more than a physician makes a diagnosis directly from a patient's symptoms. In each case there are one or more intermediate levels of abstraction at which the expert reasons. The physician repeatedly combines several symptoms into a more general, more systematic indicator (e.g., specific temperature readings over a three day period may combine into "fever slowly increasing"). Similarly, the I and W analyst may combine many reports of specific enemy activity (e.g., attack formation of regimental and divisional troops in certain key areas) into a general conclusion (e.g., "deployment of combat elements in echelon"). Such general statements are called indicators, (i.e., classes of activities which signal an important change in the threat posture of an enemy force). An indicator becomes active when it is determined that such a change has taken place. The expert can then reason from a small set of activated indicators and go back to the original data just as a check of his conclusions. An indicator may be thought of as a way of symbolically, rather than statistically, "reducing" a huge mass of incoming data.

A time sequence of indicators (i.e., expert "Artillery well forward and massed," then "Maneuver forces in pre-assault formations," and so forth)

and a collection of threats (at various levels of generalization) spanning the same time period is called a threat scenario. The first step is to select a useful set of indicators which would generate viable threats and threat scenarios and provide the basis upon which to explore the utility of tactical I and W expert systems.

Table 1 contains a sample list of attack and defense intelligence indicators. An example of a tactical threat scenario is presented in Figure 1.

SENSOR AND INTELLIGENCE DATA ORGANIZATION

This step is essentially the initial phase of the knowledge acquisition process discussed below. Based on (1) relevant intelligence documentation, (2) existing sensor capabilities, and (3) knowledge of experienced analysts, generic report categories will be developed for each indicator that can accommodate specific information pertaining to the activity levels. Figure 2 is an example of an indicator and some of its possible associated report categories.

Each report category should be further defined in terms of additional parameters. These parameters are based on the following factors:

- Expectations about observed events
- Pertinent geographical features
- Types and reliability of sensors and intelligence sources.

This defines a format in which incoming reports will arrive. Each report will have specific values given by sen-

sor and intelligence data and by knowledge about sensor characteristics, expected events, and geographical areas. An example of a report pertaining to the Increased Engineering Operations indicator as shown in Figure 2, and belonging to the road-and-bridge repair category, is given in Table 2. For each item on this list, the name of the corresponding parameter is on the left-hand side of the colon sign and the value appears on the right-hand side. In most cases, the latter is a symbolic expression rather than an actual numeric value.

KNOWLEDGE ACQUISITION

The knowledge acquisition process is probably the most critical phase in the development of an expert system. It consists of acquiring a body of knowledge adequate for achieving an expert problem-solving performance in the specific task domain. Knowledge sources include the following:

- Published material such as textbooks, documents, and so forth about the domain and its problem-solving methods.
- Examples of problem-solving instances.
- Experts' past problem-solving experience.
- Experts' knowledge relevant to problem solving.

For complex and dynamical changing task domains such as tactical I and W analysis, the last two types of knowledge play a key role in the successful development of expert systems. Thus, in addition to information extracted from documentation on military doctrine and tactics, the development of

Table 1. Typical Intelligence Indicators

ATTACK

Attack may be indicated by--

<u>ACTIVITY</u>	<u>EXPLANATION</u>
Massing of mechanized elements, tanks, artillery, and logistical support.	Areas of secondary importance are often denuded to mass maximum strength for main effort.
Deployment of combat elements (mechanized, armor, antitank) in echelon.	Normal attack formation provides for second echelon of the regiment to be located 3-6 kilometers in rear of the first echelon on line; division second echelon 6-8 kilometers in rear of first echelon; and army second echelon 15-25 kilometers in rear of first echelon.
Forward units disposed on relatively narrow fronts.	The actual attack zone of a mechanized regiment is about 4 kilometers within an assigned frontage which varies from 5 to 8 kilometers.

DEFENSE

Defense may be indicated by--

<u>ACTIVITY</u>	<u>EXPLANATION</u>
Preparation of battalion and company defense areas.	Defense is based on stubborn defense of battalion defensive areas, and counterattacks by tank heavy forces.
Extensive preparation of field fortifications.	The enemy makes extensive use of trenches, prepared positions, and overhead cover in defensive operations.
Formation of antitank strongpoints.	Antitank strongpoints are formed along logical avenues of approach for armor. These are made up of mechanized engineer, and antitank gun units with positions strengthened by mines, ditches, and other obstacles.
Attachment of additional antitank units to frontline defensive positions.	In areas where there is a serious armored threat, the enemy will concentrate as many as 25 antitank guns for every 1,000 meters of front.

Table 1. Typical Intelligence Indicators -- Continued

<u>ACTIVITY</u>	<u>EXPLANATION</u>
Preparation of alternate artillery positions.	In normal defensive operations, three positions are prepared for each firing battery.
Employment of roving artillery.	Roving guns are part of normal defensive operations.
Large tank units located in assembly areas to the rear.	Tank units are held in assembly areas for employment in counterattack roles.
Preparation and occupation of successive defense lines.	In the defense, separate and distinct defense lines are prepared and occupied.
Presence of demolitions, contaminated areas, obstacles, and minefields.	Demolitions and minefields and other obstacles are placed to cover approaches to the position.
Deployment of mechanized units on good defensive positions.	Dominating terrain that has good fields of fire and is relatively inaccessible to tanks usually is selected for a defensive position.
Dumping ammunition and engineering supplies and equipment and fortifying buildings.	Engineering tools and equipment may be used to dig trenches and to erect obstacles.
Entrenching and erecting bands of wire.	Digging of trenches and the erection of of wire indicate preparations to hold the position.

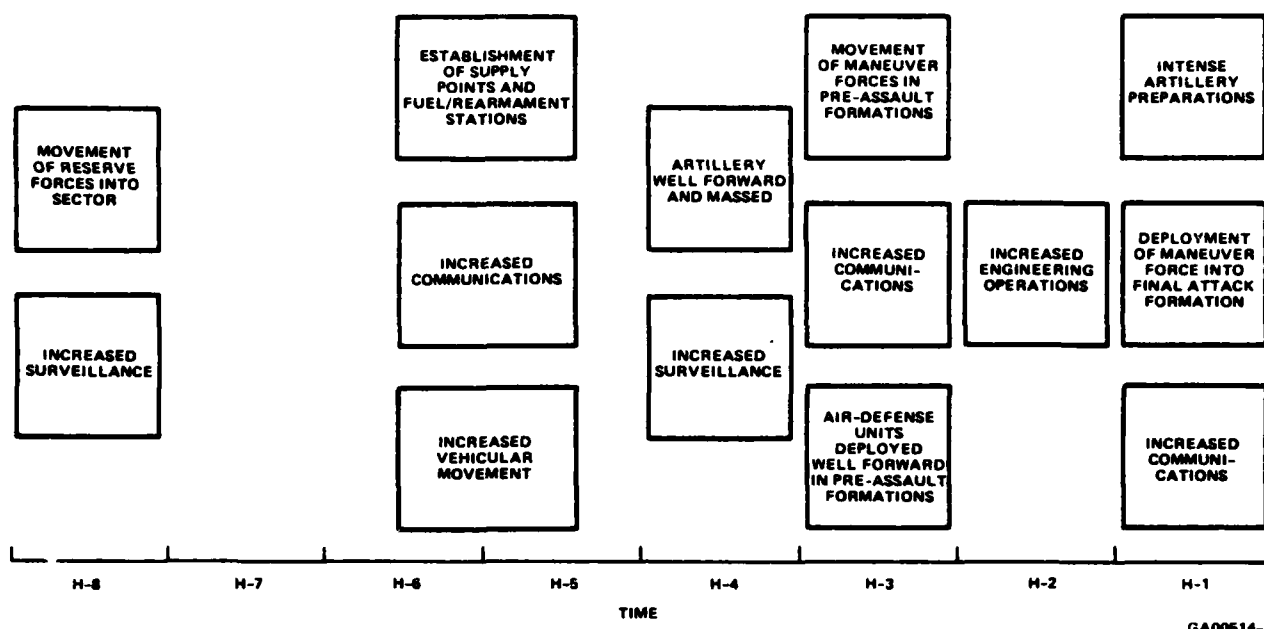


Figure 1

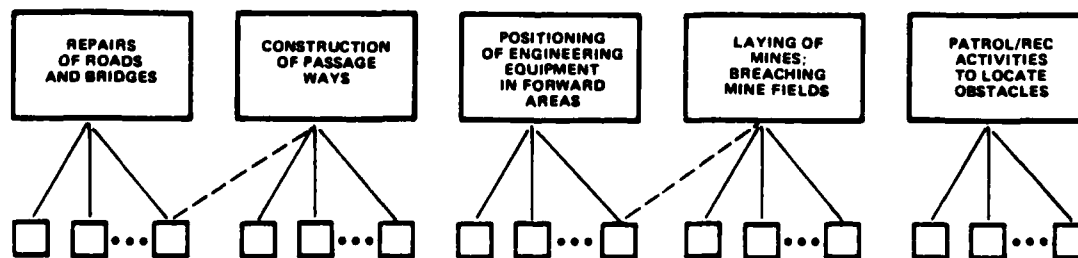


Figure 2

GA00514-3

Table 2. Example Report

Description:	Repairs of roads and bridges.
Unit ID:	Personnel wearing engineer insignia and vehicles containing distinctive engineer corps markings and distinctive engineer equipment.
Location:	Forward areas of divisional sector DS17.
Number and Size of Equipment:	Heavy road repair machinery observed in numbers greater than those organic to the division.
Terrain:	Largely forest area with trails only.
Sensor Type:	Imagery, POW, and refugee information.
Sensor Certainty:	On a scale from 0 to 100, 80 for imagery, 50 for POW, and 30 for refugee reports.
Time:	15 June 1983, 1500 hrs.
Related Indicators:	Increased Engineering Operations.

the knowledge base of the proposed prototype tactical I and W expert system will incorporate facts, procedures, and heuristics supplied by in-house and Government-designated experts. Experts will be interviewed about how they

- Know when to activate each indicator.
- Identify vital but missing pieces of information, which would heavily affect their decisions (this drives collection tasking).
- Read patterns to conclude that certain scenarios may be in progress (this provides tactical warning).

Rules* will be developed for these analytic tasks. In general, the rules will be influenced by various factors. For example, the indicator activation criteria must be based both on the types and nature of data in specific reports and on relationships among indicators. Furthermore, the level of completeness of the specific reports and the reliability of their sources will result in rules that conclude activation with varying degrees of certainty.

* By a "rule" we do not mean a rigid constraint such as a rule of a card game. Rather, we mean a flexible guide to plausible action or decision-making, a rule of good judgement and good guessing.

In particular, knowledge about indicator relationships will be important for two reasons. First, based on current indicator activity, meaningful hypotheses could be made about verifying, expecting, and predicting threat developments in the past, present, and future. Secondly, the cover and deception aspects of incoming report data could be minimized. This means that reliable rules about activating a certain indicator should be based not only on specific reports (which could potentially reflect cover and/or deception activities) but also on the relationship of that indicator to other indicators within the context of threat scenarios.

An example of a criterion for activating an indicator called "Artillery well forward and massed" is given in Table 3. It should be noted that the last two bullets of the IF portion of the table are statements about active indicators.

The knowledge acquisition process must also address tactical I and W tasks pertaining to unconventional (nuclear) battlefield environments. Of course, significant constraint on threat assessment for such environments is the unprecedented nature. Obviously, large-scale engagements with enemy forces in critical areas of the world (such as Europe) might create novel and previously untested situations. In such cases, I and W analysis would be performed largely on the basis of expert knowledge and judgements about expected capabilities of new weapons, hypothetical enemy tactics in potentially unconventional environments, and so on. Table 4 gives an example of a criterion for activating an indicator which describes likely deployment of nuclear weapons.

Another aspect of the knowledge acquisition process will be concerned with the data input problem. The objective of this element of the proposed effort would be to acquire knowledge for the following:

- Identifying from the masses of collected data those processed sensor and intelligence systems outputs that are relevant to tactical warning (for example, information about movements of units of relatively small size or the deployment of certain types of equipment may be insignificant if considered within the warning context).
- Assigning these sensors and intelligence information components to the appropriate report categories and parameters developed as a result of the data organization effort.

The successful performance of the first task will significantly improve the efficiency of analyzing large amounts of data.

The second task will contribute to the development of capabilities for automated input of data to the tactical I and W expert system.

Finally, knowledge acquisition is an iterative process as shown in Figure 3.

PROTOTYPE SYSTEM DESIGN

ESL's approach to the prototype system design will consist of the following steps:

Table 3. Example Activation Rule

IF

Within the last two hours it has been observed that

- All organic artillery batteries in the divisional sector DS7 are in place.
- They are complemented by several non-organic batteries.
- These batteries are within 1/3 or less of their maximum range behind the FLOT and in close lateral proximity to several potential key friendly targets.
- The above information is generated from an imagery sensor of highly reliability, and the sensor timeliness factor is less than 30 minutes.
- There is an increase in artillery-related communications in DS7.
- An increase of logistic build-up in DS7 has been detected.

THEN

Activate the indicator "Artillery is well forward and massed" with high probability.

Knowledge Representation

In the field of artificial intelligence, a representation of knowledge is a combination of data structures and interpretative procedures (i.e., scheme for symbolically describing part of the world). If appropriately used in a program, these structures and procedures can lead to a knowledgeable behavior. Research on knowledge representation in AI has involved the design of several types of structures for strong information in computer programs and the construction of procedures to manipulate these structures for the purpose of making inferences.

A representation method that has been used in most of the successfully developed expert systems is based on the idea of representing knowledge in the form of rules or productions. A rule is

simply a condition-action pair. IF this condition holds, THEN take this action. In Table 3, the example of an indicator activation criterion is already expressed in the form of a rule. The IF part contains the conditions or premises and the THEN part is the conclusion or action. If the conditions are satisfied, then the rule can fire, that is, its conclusion is executed by the program. The rules examine a data structure, called the context, which means to represent the current state of the world and check to see if their IF parts are true in that state, in which case the THEN parts can fire. In an I and W expert system, the context will be comprised of incoming specific sensor and intelligence reports, order of battle, geographic and military tactics knowledge reflected in the conditions of the rules and conclusions of rules that have

Table 4. Example Activation Rule

IF

In the divisional sector it has been observed within the last six hours that

- Units are dispersing.
- Troops are digging in, hasty locations are being fortified.
- There is an increase in medical units, transports, and so on.
- Special decontamination equipment is observed in enemy area.
- Enemy soldiers are observed carrying nuclear, biological, and chemical (NBC) protective equipment.
- Chemical/nuclear ammunition/delivery means are observed.
- There is an increase or notable existence of special, secure communications traffic.
- There is an increase in meteorological/target acquisition activity.
- Wind/weather conditions favor enemy use of NBC.

THEN

Activate the indicator "Enemy is likely to use NBC weapons in next 24 hours" with 50% probability if no NBC use was made sooner, otherwise the 90% probability.

Table 5. Example of a Rule for Indicator Relationships

IF

There is strong evidence that artillery is well forward and massed.

THEN

Expect increased levels of communications and engineering operations within a time period of at most two hours.

already fired. For example, the rule given in Table 5, and based on the scenario in Figure 1, has as its premise the conclusion of the rule in Table 3. The latter becomes part of the context as soon as the corresponding rule (in Table 3) has fired. Such rule changing is a convenient representation of the tactical warning process as a causal sequence of active indicators along a time line.

The above examples of rules also demonstrate another important representation feature already mentioned in the preceding section—the ability of expert systems to deal with facts and knowledge that are less than certain. Rule conclusions can be either qualified by expressions such as "likely", "highly probable", "strongly suggestive", and so on, or assigned some certainty or probabilistic numerical

factors. Both ways are acceptable to expert systems since they can perform symbolic reasoning as well as numerical computations. Furthermore, each segment of the conditional part of a rule (each expression between ANDs) can have its own certainty factor or qualifier.

Mathematical formulas or additional heuristics can combine these factors and assign a certainty value to the conclusion.

In addition to the concept of a rule, expert systems design utilizes a variety of other representation techniques. One such technique which has turned out to be particularly suitable for the development of a prototype strategic I and W analysis expert system sponsored by ESL's IR&D program is based on the idea of frames.

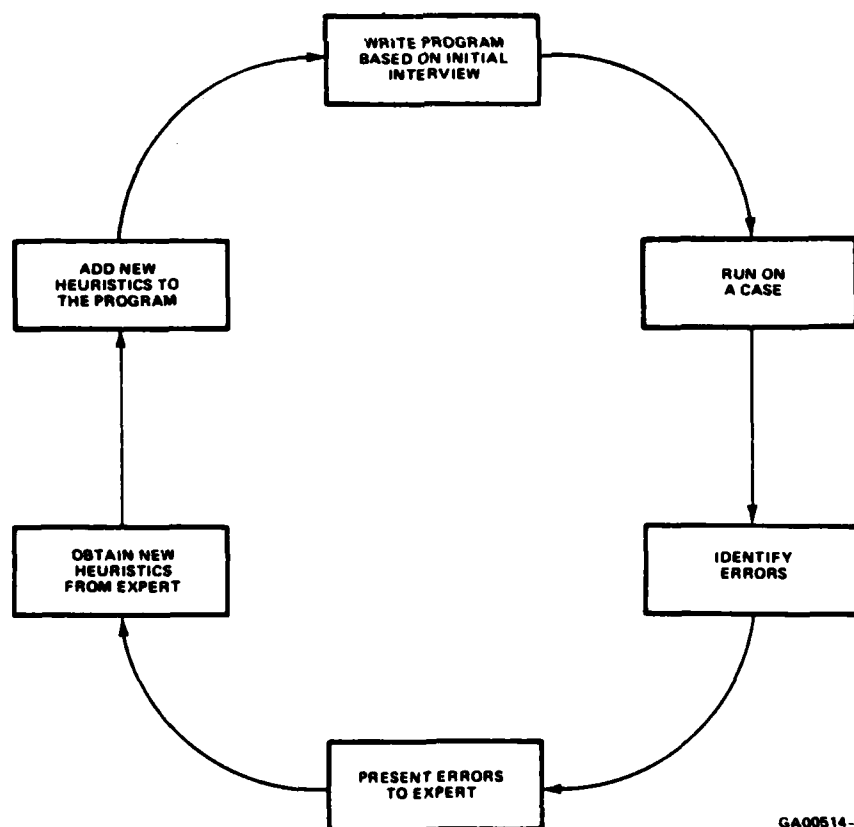


Figure 3

Frames provide a structure within which new data are interpreted in terms of concepts acquired through previous experience. The organization of this knowledge facilitates expectation driven processing, that is, looking for things that are expected based on the current context. This type of reasoning is made possible by the use of knowledge hooks, or slots (the places where knowledge fits within the larger context created by the frame). For example, the specific report in Table 2 is already represented in the form of a frame. The slots are the report parameters. They have been created on the basis of expected observations which would characterize enemy operations on repairing roads and bridges. Furthermore, the slots are also organized in terms of requirements to support indicator activation. Thus, assessments about increased engineering operations will be made on the basis of locations of road and bridge repairs, the number and size of engineering units involved, terrain information to provide further evidence of the scope of the enemy's effort, source or sources of the reports in order to determine their credibility, and so on. The slot values are determined by sensor and intelli-

gence data and by information that may be inherited from more generic frames. For instance, the information in the unit ID slot in Table 2 may be derived from a generic report category frame because of strong expectations to observe this type of personnel and vehicles.

Data input and inheritance are only part of the available slot-filling mechanisms. The values of certain types of slots can be derived as the result of rules or heuristics operating on information provided by other slots. For instance, each specific report frame may contain a slot called "unusualness". Its value may be determined on the basis of slot information similar to the one in Table 2 and would indicate the importance of the particular report in the context of tactical warning. In AI terminology, such slot filling mechanisms are called procedural attachments.

The concept of frames is a convenient representation form not only for reports but also for indicators and even the rules themselves. An example of an indicator represented as a frame is given in Table 6. Suitable slots for a rule frame would include

Table 6. A Frame-Based Representation of an Indicator

Name:	Increased engineering operations.
Instance of:	Indicator
Reports Supporting Activity:	Road and bridge repairs, construction, equipment in forward areas, clearing mines and obstacles.
Rules for Activation:	Rule 1, Rule 2,...
Related Indicators:	Artillery well forward and massed, increased surveillance,...

IF

THEN

Indicators supported

Source of this rule

Rules that might countermand this rule.

Data Input

This phase of the design effort will investigate the development of procedures which would allow the automatic frame-based generation of reports from sensor and intelligence data. The performance of this task will utilize the following factors:

- Most of the sensor and intelligence outputs are already formatted.
- The vocabulary of the sensors and intelligence messages is based on standard military terminology and, therefore, constrained within manageable limits. Thus it will not be necessary to apply natural language understanding technology which is still too complex and in many respects, unreliable.
- Frame-based structures provide a natural environment for representing sensor and intelligence data.
- The knowledge acquisition effort will strongly emphasize the development of simple heuristics which could provide mechanisms for initial data screening. For example, expert judgment rules could eliminate a number of data items based on time, location, and subjects that are likely to be covered by these items.

User Interaction

The prototype expert system will provide the basis for the development of a computer-based system for training of I and W analysts. Thus, user interaction considerations will play an important role in the design efforts. The design phase will address the development of the following user interaction capabilities.

Explanation Facility

The representation and control structure techniques, together with the tools provided by symbolic manipulation programming environments, facilitate the development of capabilities for easy access to the line of reasoning for any inference made by the system.

Modification, Change, and Addition of Knowledge

One important feature of expert systems is its modularity. Rules behave like independent pieces of knowledge. They can be added, deleted, or changed without affecting the rest of the rules, since rules communicate only by means of the context data structure and do not call each other directly. Such capabilities can substantially enhance I and W analysis training. The user can perform simulations, access the effects of changing indicator activation rules on decisions about expected threat developments and tactical warning, and so forth.

The design efforts will also identify an appropriate rule syntax which would facilitate the user's access to the program. Furthermore, the system will be designed in such a way that user-supplied changes, modifications, and additions of rules for the purposes of

tests and simulations do not permanently affect the system's knowledge base. The latter should be altered in a significant way only by experts during periodic system maintenance reviews.

Graphics

Appropriate graphics capabilities will be developed for a meaningful display of data and inferences performed by the system.

IMPLEMENTATION AND TESTING

The system implementation will be performed on ESL's Xerox-1100 Scientific Processor, within the INTERLISP-D programming environment, which has a number of special capabilities for implementing expert systems and a variety of display and graphics functions.

The system will be developed in an incremental and iterative fashion through frequent tests and refinements. For example, during the knowledge acquisition phase experts are unlikely to present all of the relevant facts and relationships for expert performance in the domain of tactical I and W analysis. Many pertinent details about domain knowledge may be supplied by the experts in the course of their examination and critique of the performance of various system modules. These procedures would also result in refining and augmenting such system capabilities as explanation, display, user interaction, and so on.

CONCLUSIONS

This paper has presented an expert systems approach to aid tactical I and W analysts in the performance of such tasks as indicator activation, information requirements collection tasking, and tactical warning. The main knowledge engineering ideas are based on our ongoing work on strategic I and W expert system development.

ACKNOWLEDGEMENT

We are greatly indebted to Mr. Eric Vieler from ESL, Incorporated, for providing his expertise as a former Army Intelligence Officer.

SYNTAX PROBLEMS WITH SPEECH RECOGNITION IN SIMULATOR TRAINING SYSTEMS

Thomas Cutler
VERBEX

The arriving age of artificial intelligence and expert systems in military operations creates the need for a better means of entering and retrieving data. Speech recognition has for some time been a candidate for such data entry but due to its embryonic status requiring the isolated word mode, it has been deemed too cumbersome and error prone to be useful. The arrival now of truly continuous speech systems, with high accuracy, rapid data entry, and interactive query capabilities should revive serious consideration of speech recognition for this purpose. One of these new systems, recently announced by Verbex, a division of Exxon Enterprises, has all the capabilities to accomplish this task. This system, called the Model 3000, was specifically designed for continuous speech. It does not have any particular limitation on the entry of long sentences except as to how long you can speak without catching your breath; our record is 100 digits in about 30 seconds. In an application on the inspection of printed circuit boards, single utterances such as "substitute resistor trim between transistor zero three lead one five and u three four pad five six" are common. This paper will endeavor to show some of the techniques used to create the grammars (syntax) to accomplish difficult speech recognition tasks, based on the capability of the Model 3000.

Speech recognition devices have been around for many years. They have not

proliferated due to lower than desirable recognition accuracy in operational use and less than friendly characteristics when interfaced with human beings. Many systems were bought and discarded as operational problems surfaced. Only those systems which filled a real need survived. Since there was no real ability to compare capabilities of systems in the manufacturer's advertising, (they all promised 99+% accuracy) the cheaper systems were purchased for application evaluation. Since really capable systems required more intensive computation and memory they were more expensive and not often purchased for this purpose. Texas Instruments ran a test of the seven systems on the market in 1981. The test was simple and under optimum conditions which never appear in operational use. The results as reported in the September 1981 issue of the IEEE Spectrum showed much lower capability than advertised for all systems except the most expensive model which was the most computationally intensive system. That one, which was one designed for continuous speech, showed 99.8% accuracy indicating that the test was too simple to judge its real capability.

Modern advances in microprocesses and lower cost memory have now made the complex systems available at a reasonable price and there is a probability that the market can fulfill its earlier promise, to establish a billion-dollar-a-year industry. Speech recognition, however, is a formidable

problem in pattern recognition since no two people say a word the same way and a single person will also not say it alike every time. The effects of physical exertion, psychological stress, what words precede or follow a given word in a string, put a considerable strain on the variances which are computed for each sound. Continuous speech systems then have an advantage over discrete word systems. They train on both discrete and strings of words building up better template models. They can also take advantage of the redundancy that is common in continuous speech. Speakers do not always pronounce all the words clearly. Those words that are mumbled are understood by the gist of a sentence. A continuous speech utterance with predetermined grammar constraints can often recognize a mumbled word by the other words in the sentence since the recognition process has a feature that explores many possible strings of words backward in time to arrive at the optimum legal sentence. For example, suppose one said, "Detect artillery in coordinates North 2.1, East 5.2" and the word coordinate was mumbled to sound like "corridor". Now corridor might be a legal utterance following "Detect artillery in" but since the coordinates "North 2.1, East 5.2" do not define a "corridor", the system would correct its first wrong recognition in its backward retrace of the phrase. This is not practical in a discrete word system where control of the active vocabulary proceeds only forward in time and the system would expect a corridor designator such as "Z 6" and not have "North" in the active vocabulary.

Other advantages of continuous speech over discrete word systems may be found in their lack of sensitivity to the beginnings and ends of words and very

significantly in the naturalness of speaking and the consequently friendly interface with people. Constant practice makes speech a particularly good man-machine interface and most of this practice is in continuous speech.

The example in grammar constraints I would now like to show is the use of speech recognition on the Army' Conduct of Five Trainer (COFT) under development by General Electric in Daytona Beach. In this system, a tank commander and a gunner are talking to each other and to a driver and a loader who are not present in the trainer. The functions of the driver and loader are handled by an instructor with a keyboard, who is also entering what targets the TC and gunner see by their spoken words and grading how well they handle the situation. This is a task of considerable difficulty since the tank team is racing the clock to move into position, lay the guns, load, complete the fire control inputs and kill the targets.

It is also desirable for the team to be able to practice without an instructor. The use of speech recognition is capable of handling this since the grammars and vocabulary are somewhat standardized for maximum efficiency and low ambiguity.

The speech recognizer has the following special problems:

1. Two people will be talking to the same recognizer although not simultaneously.
2. They will talk to each other when not giving or receiving commands.
3. There is almost no feedback to the speakers as to whether or

not their words are properly recognized.

4. There may be as many as forty commands or instructions in a single exercise which must all be understood by the recognizer to properly complete the exercise.
5. It is common to have ten or more words spoken in a continuous stream without pauses.
6. There is stress and excitement in the exercise which tends to change voice characteristics.
7. The speakers will deviate from the prescribed script so allowances must be made for these errors to keep the recognition process going.

In general, it is apparent that the accuracy requirement is high and that the variance must be chosen to allow for stress and excitement. If possible, these should be worked into the enrollment and training (the vocabulary) phase. The use of a constrained syntax also aids in recognition accuracy.

With respect to spurious talking, Verbex has included in the design a relatively reliable means of ignoring extraneous speech. This method consists of making a general non-word template which appears before legal utterances have started. This template, named the "Joker Word", absorbs all speech that is above the ambient noise level but has a lower threshold than a legal word. It can be set at a higher threshold if occasional nonrecognition of legal words is not a serious problem, or lower if some spurious insertions are not serious.

In the COFT trainer the following sequence of events occurs:

1. Tank commander (TC) and gunner view scenery through scopes
2. Targets appear
3. TC orders driver to move out and engage
4. TC designates targets
5. Gunner orders driver to stop when he sees target
6. Gunner makes fire control adjustments and fires on TC's order
7. Spotting of rounds is made by gunner and verified by TC
8. Re-engagements are designated by TC
9. After "Cease Fire" driver is ordered to back up over a hill to reduce silhouette

Some of the eight grammars in one of the exercises are shown below using the node-arc technique. The small circles are nodes when the speaker may pause momentarily or continue through. In grammar A the TC should say what is on the top line to get a perfect score..note that "sabot" is a special antitank round, "coax" is the coaxial machine gun, and "PC" stands for personnel carrier.

In Line 2 of grammar A we have entered in the capability for him to misrecognize the targets and designate the wrong weapon without halting the speech sequence. In line 3 we have allowed the TC to designate and engage only one target; the dotted line permits him to omit the words in

that arc. Line 3 also shows how the word "correction", returns him to the proper node to enter the correct designation. The system will not shift to the next grammar until it hears the word "takeover".

This is a simple representation of the many deviations which might be expected in a single grammar like; it is given only to show the techniques.

In the second figure, the first line shows how we can accommodate any of three commands: "At my command, Fire", "Fire" or "From my position". Only the words "fire" or "From my position" signal a shift to the next grammar. The second line in this figure shows how the speech train can split into two sets of arcs and end up in different nodes that exit to different follow-on grammars.

The techniques in themselves are rather simple, but some smart thinking

is required to make complex grammars that can handle reasonable deviations from the proper script.

Users tend to appreciate allowances for deviation because they can accomplish their objectives on the recognizer with fewer entries.

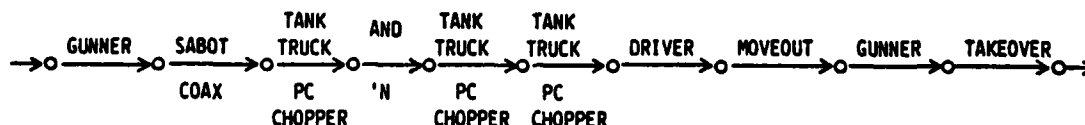
To permit the generation of new grammars by customers, Verbex has developed a special modification of its standard Model 3000 called the Speech Processor Application Development System (SPADS). The software in this system provides all the tools to create and debug new application programs with relative ease and by people who are not expert in programming. It handles application with vocabularies resident in the speech processor of up to 360 words. Several military laboratories have already ordered these machines for a variety of purposes.

GRAMMARS AND SYNTAX (1)

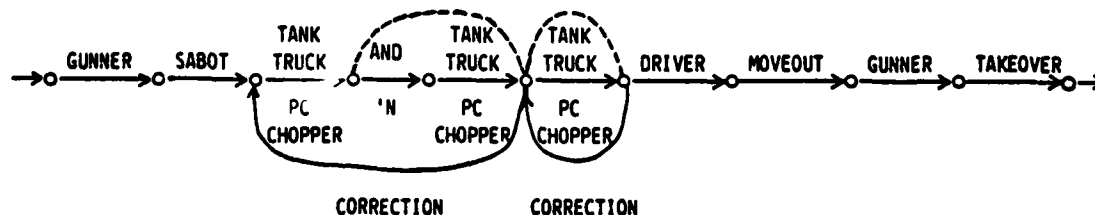
GRAMMAR A 1) AS INTENDED TO BE SPOKEN



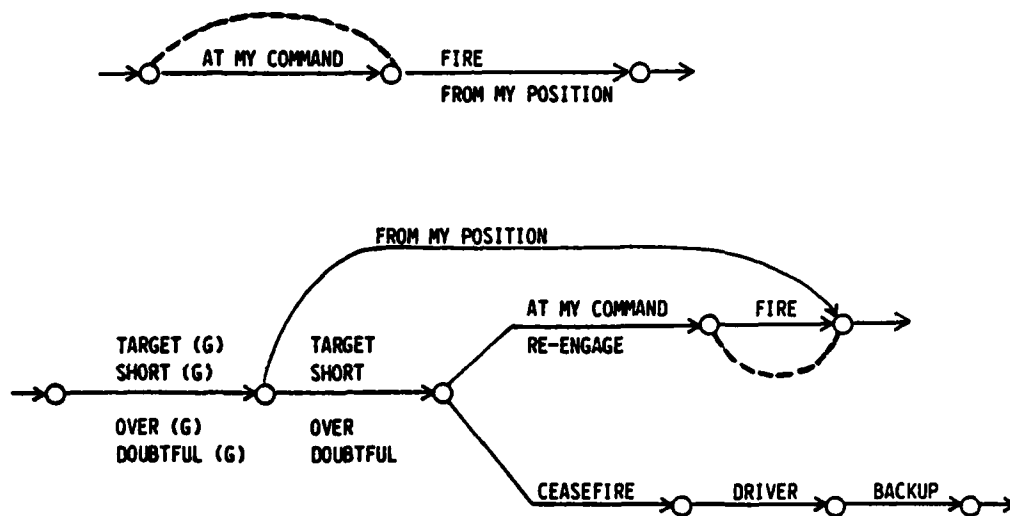
2) ALLOWANCE FOR ERRORS



3) ALLOWANCE FOR DEVIATIONS AND CORRECTIONS



GRAMMARS AND SYNTAX (2)



A MESSAGE UNDERSTANDING FRONT END FOR A KNOWLEDGE-BASED THREAT WARNING SYSTEM

Dr. Christine A. Montgomery
LOGICON

Previous automated support for threat warning applications has concentrated on the observable data manipulation operations of an intelligence analyst rather than on the complex mental operations that are most critical to analysis. There are two main reasons for this: in the first place, the cognitive operations of analysis have been a "black box" that research sponsored by ARI and INSCOM has only recently illuminated; secondly, the technology for automated assistance to cognitive activities--for example, AI in general, and knowledge engineering and natural language understanding in particular--effectively did not exist. Because of the recent developments in these two areas, it has become realistic to design and construct advanced experimental systems that can assist an intelligence analyst as a human colleague would, monitoring a threat-related situation, identifying indications of threat-related events, suggesting hypotheses concerning the nature of the threat, and performing various search, retrieval, and report generation tasks for the analyst.

The Active Information System is an experimental model of this type. As opposed to the conventional passive, user-driven, information systems constituting current information systems technology, the Active System is data- and goal-driven. It is capable of assuming the initiative in analyzing incoming data and in

communicating with the analyst-user based on prestored goals representing the features of the threat and procedures that monitor incoming data and allow the system to introspect about its own contents concerning the threat situation. The Active System is comprised of three major components: the component containing the Active/Introspective Processor and associated data and knowledge bases, the user interface, and a data acquisition component.

The last of these is a front end that reads the incoming electrical message traffic as an analyst would, distilling out items of interest and transforming these into data base elements for the Active/Introspective Processor to analyze and store. This component contains syntactic and semantic subcomponents that analyze the natural language text of the incoming messages in terms of an inventory of frame structures representing knowledge about entities and events in the domain of space and missile (S&M) activities. A testbed system called MATRES is written in the Prolog language and runs on the PDP-11/70 under the UNIX operating system.

This presentation will describe MATRES and the Active System in terms of an S&M threat warning application, focusing on aspects of the architecture that appear generalizable to a variety of threat warning situations.

TECHNOLOGICAL ASSESSMENT OF FUTURE BATTLEFIELD ROBOTIC APPLICATIONS

Barry J. Brownstein, John J. Reidy, and G. Frederick Renner
Battelle Columbus Laboratories

INTRODUCTION

The process of applying robotics to the battlefield appears to be, to a certain degree, concept driven. That is, a concept for a robotic system is generated, then its value to the military is assessed. Alternatively, detailed applications may be specified, then the concepts are generated. Neither approach represents the thorough, integrated procedure necessary for the optimal application of robotics by the military. A different approach is to examine the military application of robotics from a technological viewpoint. In this paper,

we have considered two aspects of robotics technology: the basic technologies involved in the application of robotics, and the application process itself.

As the first step in discussing robotics technology, we must define the terms robot and artificial intelligence. Although numerous definitions of both are available, we will use those of the Army Science Board (ASB). The ASB defines artificial intelligence thus: "A programmable machine exhibits artificial intelligence if it can incorporate abstraction and interpretation into information processing and make decisions at a level of sophistication that would be considered intelligent in humans." To the ASB, a robot is essentially the embodiment of artificial intelligence in a mechanical system. Or, as the ASB states, a robot is "a programmable machine that displays cognitive behavior and performs mechanical and manipulative functions

similar to those performed by humans."

Several potential applications of robotics to military tasks are in various stages of conceptual development. These are listed in Table 1. Some applications are weapons oriented, such as intelligent mines, robotic flame throwers, and sentries. Other applications are oriented toward combat support and reconnaissance tasks. Some are very large systems, others are quite small. This list also includes applications that are not truly robotic at all. For instance, the brigade planning aid is an artificial intelligence system, packaged to help field commanders make quick decisions. And a maintenance training aid may or may not be robotic. For many military applications, the distinction between robot and artificial intelligence has become blurred; few examples of applications of robotics to military purposes do not include some degree of artificial intelligence.

Table 2 lists some, but not all, of the technologies required for the robotics applications listed in Table 1. Some technologies are closely allied with artificial intelligence, such as vision, guidance, and sensor fusion. Others are related to "mainstream" digital technology sensor systems, such as computer hardware and communications (especially of the secure type). Still other technologies are hardware oriented, such as energy storage systems, mobile platforms, and lightweight structures. Some of these technologies are being pursued primarily because of robotic

TABLE 1. SOME POTENTIAL APPLICATIONS

Intelligent Mines	Robotic Smoke Generator
Chemical RPV	Robotic Flame Thrower
NBC Recon	Brigade Planning Aid
Smart EOD Robot	Gunner's Aid
Missile and Rocket Loader	Maintenance Training Aid
Rapid Excavator	Tactical Reconnaissance Robot
Countermine Vehicle	Approach Sentry
Forward Ammo Handler	Light Fighting Sentry
Container Handler	Street Walker Robot

TABLE 2. SOME ENABLING TECHNOLOGIES

Vision	Tactile Sensing
Guidance	Advanced Computer Hardware
Navigation	Mobile Platforms
Sensor Fusion	Energy Storage Systems
Communication	Lightweight Structures
Planning	Expert Systems
Human Factors	Information Representation
Actuators	

applications. Different applications make different demands on the development efforts for the enabling technologies.

The potential applications of military robotics are fairly diverse, as are the enabling technologies. Ideally, the technologies would be systematized to emphasize the ones that make many things possible. However, the most important thing is to make sure that the driving factors that are most important militarily have the greatest impact on setting the research agenda. In other words, we must concentrate on enabling technologies for those applications in which we are most interested.

WHICH APPLICATIONS ARE IMPORTANT?

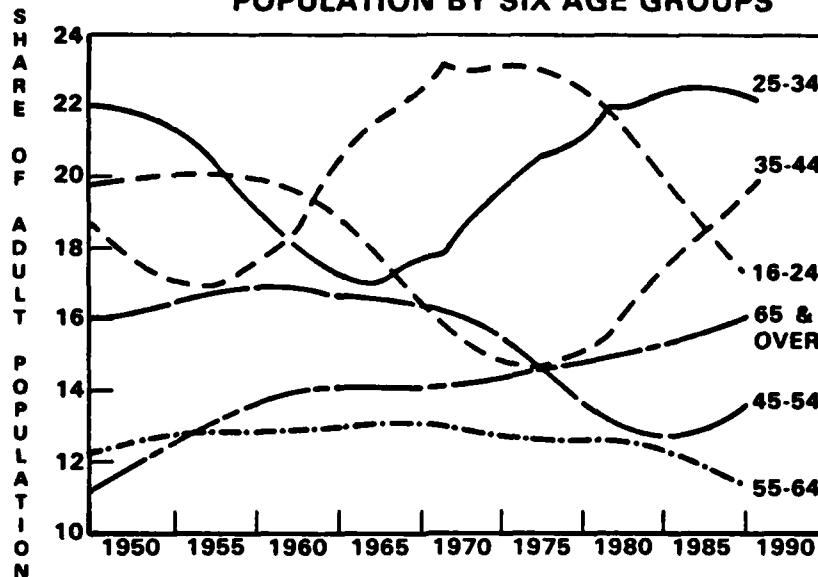
Battlefield robotics has two basic driving factors. The first is the

substitution of high technology for people. The second is the overcoming of mismatch in mass. The first factor can be expressed in a number of ways, such as "reduced manpower intensiveness" or "increased tooth-to-tail ratio". These expressions acknowledge the fact that a battlefield is really no place for a human being.

The second factor reflects the large amounts of money required to field an army large enough to match a potential adversary's. We have finite quantities of personnel and money, which we would rather not have destroyed in combat, for reasons both humanitarian and economic.

Figure 1, which illustrates trends in the composition of the population as a function of age group, reveals one coming problem; the declining availability of potential military

FIGURE 1. AGE COMPOSITION OF THE ADULT POPULATION BY SIX AGE GROUPS



personnel. This figure shows that the number of people in the 16 to 24 age group, and to some extent in the 25 to 34 age group, will be declining toward the end of this century. Thus, it will be increasingly difficult to get the number of soldiers that might be needed, especially if our society does not change its conscription techniques.

From the economic standpoint, many demands will arise between now and the end of the century. For instance, a great deal of money will be required to convert present manufacturing techniques into computerized processes. Money will also be required for the service and the information sectors of our economy. In short, as Table 3 shows, our economic system is going through changes that will require massive capital expenditures. And capital expended on warfare or on building large standing armies is diverted from the economic sectors. Each soldier trained from the army represents a significant investment; therefore, the substitution of robots

for soldiers will also be driven by future economic considerations.

Based on these long-term considerations, we have developed priorities for applying robotics to the battlefield. First, robotics must replace people in hazardous jobs, such as combat, since those people can be killed. Second, robotics should replace people in military jobs that may not be hazardous, such as in logistics, to decrease the overall investment in the armed forces. Third, robotics should be used in those applications, particularly in combat, that can overcome the disadvantage in numbers of personnel. Considering these priorities, we will next look at potential applications and what, in terms of technology, is required to bring them to actuality.

TECHNICAL PERSPECTIVE

Future battlefield applications of robotics can be evaluated on the basis of levels of complexity of the

TABLE 3. PER WORKER INVESTMENT

Current Levels of Per Capita Capital Investment by Economic Sector		Projected Per Capita Capital Investment By 1990	Projected Expendi- tures For Increased Automation Through 1990
Agricultural Sector	\$55,000-\$85,000	\$55,000-\$85,000	-0-
Industrial Sector	\$25,000-\$35,000	\$35,000-\$45,000	\$180 Billion
Service Sector	\$5,500-\$6,500	\$7,500-\$8,500	\$70 Billion
Information Sector	\$2,000-\$2,500	\$8,000-\$8,500	\$380 Billion
			\$610 Billion Total

applications. Each level requires certain technological attributes. And each higher level will require new attributes in addition to many of the attributes of the lower levels.

Level 1: Computation, Storage, and Retrieval

The first level applications include decision support systems and other such concepts that require improvements in computer technology, and in information storage and retrieval. We define a "decision support system" as a convenient, rapidly accessible source of information that provides assistance in analyzing or planning military operations. It is a very high speed source of reference information that can help a battlefield commander make decisions that he may not have time to make using a more traditional approach. An example of such a system is a tactical decision aid that is being developed at Battelle. This aid will assist wing commanders in selecting sensors and weapons on the basis of particular targets to be attacked.

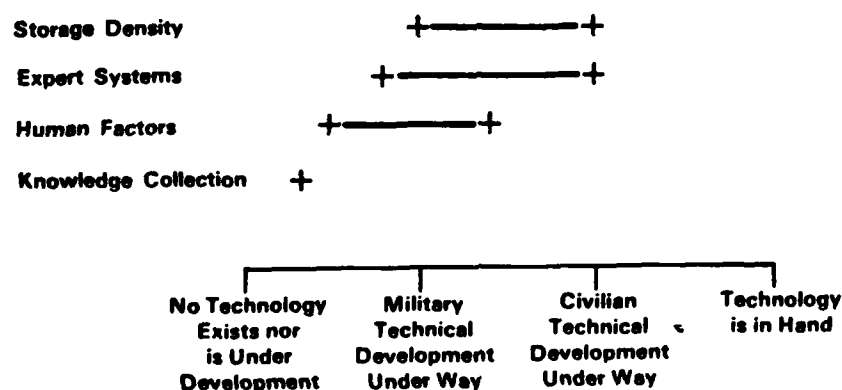
Decision support systems require a number of technologies: storage density, expert systems (that is,

artificial intelligence geared towards representing and utilizing expertise from human beings), human factors (the ability to provide a friendly interface between machines and users), and the ability to collect knowledge so that it can be used systematically.

Figure 2 displays these four technologies against a scale that reflects their relative level of development. At the low end of the scale, the technology does not yet exist; at the high end the technology is off-the-shelf; and in the middle the technology is either under development for military applications or for civilian uses. Development for civilian use is slightly better than development for military applications from the standpoint of minimum risk.

We see from this figure that storage density and expert systems are being worked on quite broadly. In the human factors area, the need for development is somewhat greater; that is, less directly applicable technology exists. Above all, however, improvements are needed in knowledge collection, which is crucial for this particular application. We have to be able to codify information such that it can be used by an automated system. That information has to be obtained from military

FIGURE 2. LEVEL 1—COMPUTATION, STORAGE, AND RETRIEVAL



experts, panels of skilled individuals, and so on. Thus, knowledge collection is a very important area of research, needed to make a reality of decision support systems that would be of use on the battlefield.

Three observations can be made with respect to Level 1 applications. The first is that the application is not of primary importance with respect to the selection criteria we set forth, although it may assist a commander in more efficiently utilizing his human resources. The second is that most of the technologies involved in making decision-support systems a reality for military applications are being worked on for other reasons. Finally, the major need is to understand the knowledge base involved; that is, to know what information a battlefield commander would have to have at his disposal in order to be more effective in the field.

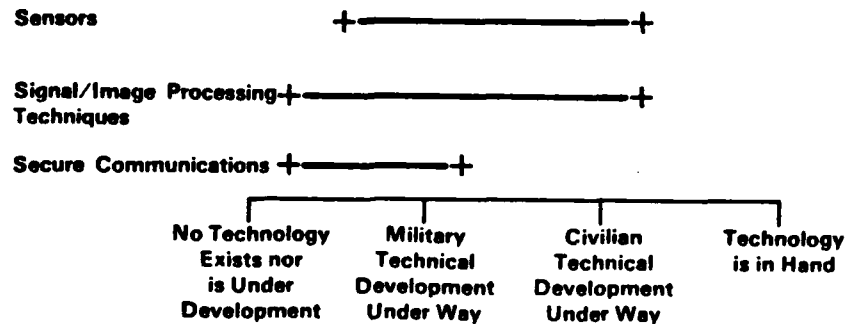
Level 2: Sensing, Signal Processing, and Communications

Level 2 includes such applications as remote information collectors, which

are capable of sensing, processing the information received by the sensors, and sending that information to some other location so that decisions based upon the information can be made. One example is a remote monitoring device for detecting vehicles. Another example is a package capable of sensing nuclear, biological, or chemical hazards, recognizing them, and recording the information so that the appropriate action can be taken. Key technology needs for such systems fall into the areas of sensors, signal and image processing techniques, and secure communications. The status of these technologies are summarized in Figure 3.

Sensors represent a very broad technology. In some cases, sensors are available off the shelf; in other cases, depending on the phenomena to be sensed, technology is at an infant stage. Signal and image processing techniques likewise cover broad areas. In some areas, we have very well-developed capabilities in processing data. In others, our capabilities are not as well-developed; for example, in visually recognizing certain objects as threats. The area of secure communi-

FIGURE 3. LEVEL 2—SENSING, SIGNAL PROCESSING, AND COMMUNICATIONS



cations, under intensive development for military applications, is the key to many important AI/Robotics applications.

Two observations can be made with respect to Level 2 applications. The first is that the application itself does spare people from hazardous jobs. Hence, the application is valid with respect to our overall requirements. The second observation is that some of the technology is available, but real gaps exist, particularly with respect to signal processing.

Level 3: Feature Extraction and Inference

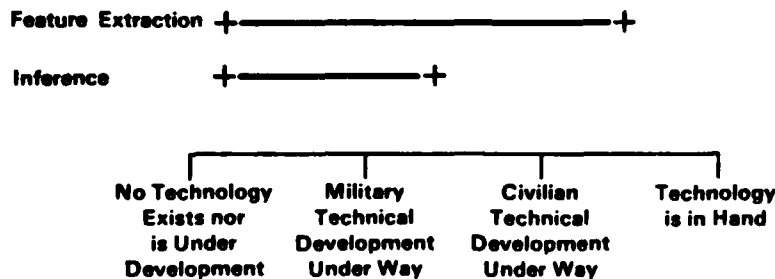
Level 3 includes such applications as smart reconnaissance aids. These aids not only can sense information, but they can recognize features in the data that it acquires and it can infer from those features what the threat might be. That is, it makes higher-level decisions than a remote sensing package does. This aid can be defined

as a device capable of filtering sensor data and, working with incomplete information, recognizing the presence of specific situations, based on sound, vision, or whatever other phenomenon is useful.

Figure 4 illustrates the state of the art of the key technologies. Feature extraction is the processing of sensor data to the point where specific signatures can be identified and recognized. This is a very broad area that is being addressed in the civilian sector for applications such as medical diagnosis and nondestructive testing. Inference is actually a broad series of techniques that interprets information to build a "mental image" of the situation being faced, even though all of the facets of that environment have not been observed directly. This also is an area that is under development, though primarily for military applications.

Two observations can be made on Level 3 applications. First, such applications are applicable to the selection criteria because they take

FIGURE 4. LEVEL 3—FEATURE EXTRACTION AND INFERENCE



the place of people in hazardous situations. Second, many technology gaps exist in this level, some of which will be very difficult to fill.

Level 4: Sensor Fusion and Decision Making

Level 4 is characterized by such systems as remote sentries and onboard diagnosticians. Here, the traits of interest are sensor fusion and decision making. Sensor fusion refers to the integration of data from a variety of sensors and the subsequent development of high level conclusions regarding the source of the stimuli. That is, if it looks like a tank, sounds like a tank, exhibits the electromagnetic interference of a tank engine, and so forth, then it probably is a tank. Such systems are capable of collecting a variety of data, of synthesizing information by analyzing and combining the data collected, and of making decisions as to the actual situation based on this information. The decisions have to be made in contexts that differ greatly from application to application.

There are two main technological needs at this level: information representation and reasoning. As

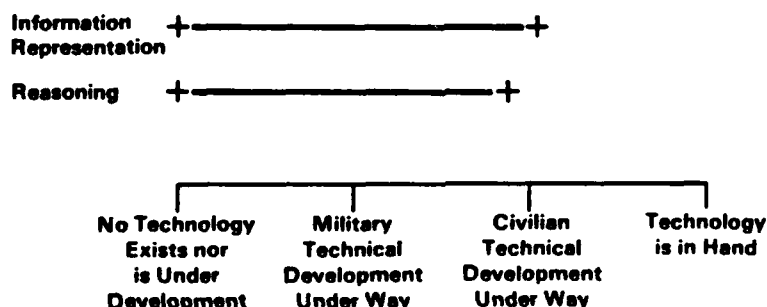
shown in Figure 5, both are subjects of a great deal of basic research. Information representation relates to how information can be represented and organized in a computer so that the kind of symbolic processing that is required can be conducted. Information representation techniques currently being applied are often restrictive and specialized due to computational limitations; hence, they are not as flexible as more generic schemes would be. Reasoning approaches as will need to be developed to deal generically with the information.

Three observations can be made with respect to Level 4. First, of our two examples, the remote sentry better fits our criteria than the diagnostician, but both are quite useful. Second, some technology is available for application. Third, to address the needs in the near future, it will be necessary to work on specific applications, rather than concentrate on generic solutions.

Level 5: Actuation

Level 5 encompasses, or adds actuation to, the suite of characteristics that we have described thus

FIGURE 5. LEVEL 4—SENSOR FUSION AND DECISION MAKING



far. It is exemplified by the concept of an autonomous weapon. An autonomous weapon can be defined as a device capable of identifying, locating, and locking on a target and firing. An example of such a system is the Navy Phalanx, which is a ship mounted, last ditch, air defense device that detects and tracks a large number of targets, categorizes threats, and launches weapons.

The principal difference between Level 5 and Level 4 systems is the inclusion of actuation. As shown in Figure 6, actuation is a very broad area. For advanced weapon system concepts, actuators do not exist. A second area in the technology is that of intelligent control. That is, the real time aspects of analyzing sensed information and directing actuation to take effective action. This is a technology that is not very far advanced.

Three observations can be made about Level 5. First, the technology needs tend to be application dependent, just as the actuator or the weapon system are application dependent. Second, energy-efficient controlled actuation could be improved. Third, the Level 5

applications, the autonomous weapons, are applicable to our selection criteria in that they enable military operations without as large an expenditure of manpower.

Level 6: Mobility

Level 6, the final level to be considered, incorporates mobility with the other aspects to produce devices of which an example is a "robotic warrior." A robotic warrior is a mobile, autonomous system, capable of carrying out a variety of military operations. Such systems could be designed for different media: air, land, sea, and undersea. Each would be capable of identifying threats and conducting military operations. Such systems may have to be "briefed" electronically prior to the start of an operation so that an excessive amount of onboard intelligence would not be required. This level clearly is the ultimate high payoff area, but it is an area in which a great many needs exist at present, and one in which progress will be incremental.

Some of the technology needs are shown in Figure 7. These relate to the areas of guidance, navigation, and

FIGURE 6. LEVEL 5—ACTUATION

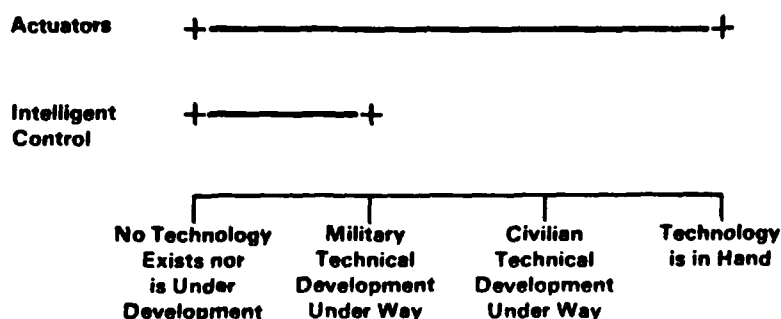
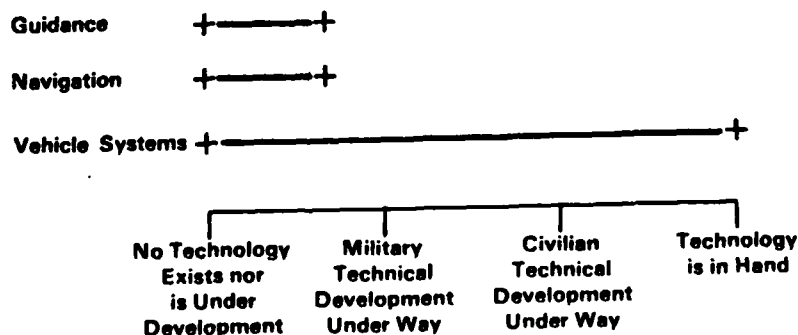


FIGURE 7. LEVEL 6—MOBILITY



vehicle systems in general. Guidance and navigation can be considered together in the sense that navigation is the ability to control one's path over a large scale while guidance is the ability to avoid short-range obstacles and to maintain one's attitude. The technology in both of these areas is at an infant stage, compared with what is required by the robotic applications. Ultimately robotic warriors will require guidance and navigation systems that are wholly self-contained and not dependent on satellite position

systems or the like in order to be truly reliable. Although most approaches that would permit these robotic systems to navigate are related to the various mapping data bases currently under development, a great deal of basic work needs to be done.

Vehicle systems encompass a large area in that a robotic warrior may be a fully autonomous version of an existing military vehicle or it may be an entirely new vehicle system, which would have to be developed, possibly

requiring different energy systems and propulsion systems.

Two observations can be made on Level 6 application. First, this is the ultimate payoff area; it is the complete removal of people from hazardous situations. Second, significant research needs exist, some of which can be handled generically, such as autonomous approaches to guidance and navigation. Others need to be considered specifically, particularly in the case of vehicle systems and some of the supporting technologies.

APPLICATIONS PERSPECTIVE

For the most part, current efforts to develop robotic military systems are driven by selected applications which have specific technology needs. This is an important first step in the implementation of robotics in the military, but it is a near-term approach. By assessing robotic battlefield applications by generic levels of technical complexity, a different perspective on future applications of robotics is achieved. The generic applications levels reviewed in this paper have identified basic technology gaps that will impact the future implementation of robotics. Both of these approaches however, tend to interface robotics into current military strategies.

Initial industrial applications of robotics were similar in approach. Robotics were assessed with regard to their application to existing processes and equipment. The benefits of applying robots were increased

substantially, however, in those cases where it was possible to modify the environment to take advantage of all of the robot capabilities. This involved such efforts as the design of work stations around the robot and the design of the product to facilitate robotic fabrication.

The current applications of robots are, without a doubt, important to the upgrading of our military forces. Not only do they increase the efficiencies of current systems, but they provide an initial mechanism for interaction between military strategy and robotics. The technologies developed for each application also impact future applications. However, as in the industrial applications, military application of robotics, particularly in the battlefield, should be assessed in the long-term. Current military strategies are based on the capabilities of the current soldiers. Implementing robots to replace soldiers in roles that have been designed around the soldier should be considered a short-term strategy.

Assessing future military strategies now in terms of both current and potential capabilities of robotics would allow the development of systems that fully utilize the capabilities of robots. Removing the soldier from the tank drastically changes the system requirements for the tank. This assessment should be a joint effort of both military strategists and technical experts in robotics and associated fields. This interaction would permit the evolution of concepts for applying robots directly to military needs, without preconceived biases based on the utilization of soldiers.

CONCLUSIONS

Projected shrinking of available manpower and funds for the military over the next twenty years will drive the military application robotics. The primary criteria for the application of robotics should be:

- 1) Replace soldiers in hazardous jobs;
- 2) Replace soldiers in non-hazardous jobs;
- 3) Increase the effectiveness of the individual soldier.

Current military applications of robotics, driven by specific identifiable needs, are intended to assist and support existing manpower. Future applications must be directed toward replacement of the soldier with fully autonomous systems.

Technologically, the military application of robotics can be classified by level of complexity, ranging from decision support aids to

robot warriors. Each of these generic levels can be assessed with regard to their primary technologies, technology gaps and their potential suitability to a future criteria of soldier-replacement. Such an analysis indicates a need for future R & D in such areas as knowledge collection, signal and image processing, feature extraction, inference, information representation, reasoning and mobility. Basic research efforts in these areas will be necessary to effectively implement robotics in the battlefield.

Finally, battlefield robotic systems can have a significant impact on our military strategies. The true value of automation in the industrial setting is not realized until the design of the product reflects how it will be manufactured. In a similar manner, the application of robots on the battlefield will be enhanced by analyzing the missions in terms of robot capabilities, not in terms of the capability of robots to imitate soldiers.

AD P03033

AUTONOMOUS VEHICLE CONTROL USING AI TECHNIQUES

D. Keirse, J. Mitchell, B. Bullock, T. Nussmeier, D. Tseng
Hughes Research Laboratory

ABSTRACT

A review of early work on a project to develop autonomous vehicle control technology is presented. The primary goal of this effort is the development of a generic capability that can be specialized to a wide range of DoD applications. The emphasis in this project is development of the fundamental AI-based technology required by autonomous systems and the implementation of a testbed environment to evaluate and demonstrate the system capabilities.

INTRODUCTION

A project to apply AI techniques for autonomous vehicle control has been under way at the Hughes Research Laboratory for the past several years. The primary goal of this project is to develop a general autonomous system "black box" that can be used for vehicle control. Other applications of the resulting technology range from complex process control to strategic command and control. The distinguishing characteristic of the type of control required for these applications is knowledge. Traditional control system methodology provides a framework for dealing with control laws that can be expressed in terms of numerical functions. For autonomous systems, the control laws of primary interest extend beyond mathematical functions, and require a body of knowledge to describe. Traditionally, this knowledge has resided only in the system designer or human expert familiar with the problem. One of the primary reasons for choosing the

problem of vehicle control for an initial project focus, over other possibilities, was that the requirements of the final system could be easily described to an observer, and the observer could easily understand and evaluate the system performance. However, it has been found that familiarity is not the same as understanding. Consequently, the body of knowledge actually required for successful vehicle control is larger and more complex than one would first suspect.

While knowledge is the ingredient, or content, that must be provided to make a system autonomous, a description of the system alone does not provide sufficient understanding of the form of the desired autonomy. For vehicle control, we have defined five system characteristics needed to achieve autonomy: 1) accept an initial model of its environment, 2) accept a description of its tasks, 3) plan intelligent decisions to perform the task, 4) accept information about environment changes from a sensor

understanding system, and 5) replan the action sequence when unexpected situations occur. An analysis of the tasks that a vehicle could be called on to perform shows that they fall into two distinct categories. The first is vehicle navigation. This is the task of getting a vehicle from one place to another in a reasonable time, without falling into holes, etc. The second is mission task control. This involves the collection of specific functional tasks that the vehicle can be called on to carry out, assuming that it can successfully handle the navigation task. Results reported here are limited to investigating the navigation task.

AUTONOMOUS SYSTEMS

Four distinguishing capabilities have been identified as general goals of autonomous system technology. These systems require fewer messages transmitted to and from the system to perform a given task. Autonomous systems are more capable, implying that they can perform nontrivial tasks. They are more dependable, implying they can succeed at what they start to do, even under changing conditions. Finally they are more distributable, being able to share their task load and pool data when necessary. It has been our experience that there is a growing requirement with DoD for systems with these autonomous capabilities. Analysis of these requirements shows that the practicality of autonomous systems is frequently driven by several related factors—the need for near real-time response rates in applications involving a high level of required functionality, the need to deal with a rapid growth of sensor or information data rates and volumes, and the need to provide a high level of

performance in a variety of possible situations.

For industrial automation, robotics, and software and hardware design tools, a great deal of work on autonomous and semi-autonomous systems has been implemented where cost effectiveness is the issue. However, there has been much less work toward the development of highly capable, totally autonomous systems aimed at DoD applications. This has been due to a strong resistance to totally autonomous systems that meet these requirements, and this attitude will prevail until one or more of the above mentioned factors proves its cost-effectiveness or increased capability over the existing manual systems. This project is directed toward the development of this type of capable autonomous system technology.

SYSTEM ORGANIZATION

The general organization of the class of autonomous systems under development is shown in Figure 1. The input to this system is a collection of symbolic descriptors that have been attached to objects in the systems sensor environment. The sensor understanding systems needed to produce the descriptors are not the subject of this project, but have been under development on other programs for over ten years at Hughes (1-3). The autonomous system consists of two major modules, one for situation assessment and one for action planning. The assessment module acts as a passive observer, using knowledge about object characteristics and interrelationships to understand the situation presented to the system at a given instant. The action planner

plays an active role. It uses knowledge about the current situation, object characteristics and capabilities, the characteristics and capabilities of the autonomous system itself, and one or more tasks to be carried out. From this knowledge it plans the sequence of actions required to accomplish the tasks. Together the situation assessment and action planner modules can provide information or control messages in response to the following environment/task related questions: what, where, when, who and why. This basic system organization metaphor can be recursively applied down to the lowest level symbols (4).

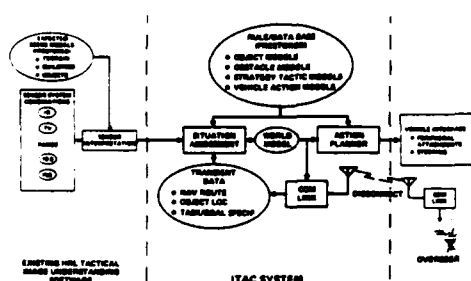


Figure 1.

SITUATION ASSESSMENT AND ACTION PLANNING

In the early phase of this project the situation assessment and action planning modules have been developed independently. Both have a common basis, however, in being knowledge based and in relying strongly on the notion of scripts. This orientation had its origin in earlier work in our group that successfully used script

representations for natural language processing of tactical Navy messages (5) and rule based systems for vision system control (6, 7).

The use of knowledge in these modules makes the autonomous systems being developed distinct from work in traditional industrial automation/robotics and earlier experiments with robot problem solving. The current emphasis on cost effective industrial robotics solutions has led to systems that have little or no feedback. In addition, they typically do not explicitly represent knowledge about all their goals, actions, and results of actions. Not modeling the real world symbolically leads to limited behavior and inability to recover from unexpected failures of action. The early pioneering work on robot problem solving was traditionally problem solver based rather than knowledge based. A serious problem with systems in this early generation, principally SRI's SHAKEY based on the STRIPS paradigm (8), was the limited class of problems they could solve due to representational inadequacy and processing time. These systems took a large amount of time to reason about even simple tasks. Later perspective has shown that a major problem stemmed from the robot being knowledge-poor, which forced the system to rely on highly inefficient general problem solving techniques.

The use of knowledge based techniques overcomes many of the limitations of the limited industrial systems now being developed and the earlier robot problem solving work. The knowledge-based problem solving approach recognizes that most behavior is stereotypical. In the system described here, three types of stereotypical knowledge representations have been used: special problem solvers, scripts, and domain-specific production rules.

SPECIAL PROBLEM SOLVERS

Certain tasks that a system must perform are solvable by an algorithm or a heuristically guided search. The element of knowledge that is important beyond the traditional algorithm, however, is knowing when to make best use of the algorithm in the form of procedural attachment. An example of an expert that requires an algorithmic solution is the problem of path-planning in vehicle navigation. Thus far, four different types of path experts have been found necessary and implemented: Shortest-path, Hide, Lost-path, and Feasible-path. All of these problem solvers make use of traditional dynamic programming methods augmented by geometric reasoning heuristics.

The shortest-path problem solver produces the optimal shortest path between two points through a series of arbitrary obstacles. The obstacles can be polygons of any complexity. Also, an optimal path problem solver produces routes through a DMA (Defense Mapping Agency) database map produced by DMA for limited areas in response to specific Army simulation. This path problem solver uses the weighted values at each point to determine optimality, as shown in Figure 2. The Hide path-planner produces a path that minimizes the distance where a vehicle is exposed to a threat it is approaching. Both Shortest-path and Hide assume perfect information of the area. Lost-path produces a path that will explore the area. That is, if the path is being traversed, then every observable point in the area will be observed. The algorithm also constructs a map of the area as the path is traversed. Feasible-path produces a path between two points using heuristic measures. The heuristic only uses the

information gathered or what the vehicle can presently see and attempts to go to the closest point to the destination, but will back up if that route is blocked.



Figure 2.

SCRIPT BASED PROBLEM SOLVERS

For unspecified but uniform problems, like path planning, a special algorithmic-like problem solver is needed. There are also specific, often recurring, problems that have simple solutions, but that no general problem solver can easily find. In that case it is easier to supply the robot with precanned plans to solve these problems.

A script is a symbolic representation of a stereotypical sequence of events. Scripts can be used extensively because most behavior is stereotypical. For example, a simple obstacle avoidance script is:

AVOIDLEFT(org,m,n,dest)

1. (BACK m); BACK UP A BIT
2. (LEFT 90); TURN LEFT
3. (FORWARD n);
MOVE FORWARD A BIT
4. (RIGHT f(org,m,n,dest));
TURN TOWARD DESTINATION
5. (FORWARD g(org,n,n,dest));
MOVE TOWARD DESTINATION

Suppose that the task is to go from point A to point B. The path planner plans a path around the known obstacle. However, there is an unknown obstacle that will block the planned route. One solution is to go around the new obstacle. Although it could be done, there is no point in deriving these steps from general principles. Instead, there should be a stored plan that can be used to repair or to patch an unsuccessful plan for going from one place to another. The execution of a script is straightforward and fast when it works.

DOMAIN SPECIFIC PRODUCTION RULES

There will be many instances when a script based solution may not quite work. In that case rule based knowledge can be used to help provide a fix. The following production rule would invoke the script.

```
IF (FAIL (MOVE org,dest))
(FAIL (FORWARD dist) movedist)
THEN (AVOIDLEFT h(org, movedist,
dest), 5, 10, dest)
```

The problem arises when the script fails. For example, if going from point A to B and an unknown obstacle was encountered, then the original plan would fail and the script fix would fail (see Figure 2). Another rule could fix the fix. An English paraphrase would be:

RULE 10: IF AVOIDLEFT failed

THEN 1. retrace movement of
AVOIDLEFT

2. AVOIDRIGHT

At some point there has to be a rule when to give up.

RULE 15: IF AVOIDLEFT
AVOIDRIGHT failed

THEN (FAIL (MOVE o,d))

SCRIPT INTERPRETER

The production system must keep track of script execution so the information needed to fix a plan exists when a failure occurs. In this project, the script interpreter is written as production rules. The production system is similar to Hendrix's system (9) and CONCUR (10). The productions can represent states and processes because the rules have durations. A production is broken into six parts: three conditions and three actions. For each condition there is a corresponding action. The first condition, the initial condition, activates the rule. The initial actions are performed when the rule is activated. The second condition, the continuing condition, keeps the rule

activated until the condition is not met. When the condition is not true, then an action, the discontinuing action, is performed. This represents the normal termination of a state or process. The third condition, the terminating condition, terminates the rule when true, and the terminating action is performed. This represents an abnormal termination of the state or process.

The execution of a script is simple: execute each step one at a time. However, the trick is making sure that each step is accomplished and if it fails, then leaving enough information for other rules to either propagate failure or fix the problem. A production, that represents the state of the script, can monitor the status of the script. The following is a production rule designed to do this:

RULE2: SCRIPT-STEPPER

INITIAL CONDITION:

IF doing a script S,
A is an action of S,
A is step N of script S,
the goal is to do step N of script S.

INITIAL ACTION:

THEN ASSERT doing script S step N
ASSERT A
DELETE goal to do step N of script S

CONTINUING CONDITION:

IF A is asserted

DISCONTINUING ACTION:

THEN DELETE doing script S step N
ASSERT goal to do step N+1 of script S

TERMINATING CONDITION:

IF A fails

TERMINATING ACTION:

THEN ASSERT do script S step N failed
DELETE doing script S step N

There are other rules which initiate and terminate the script, which are similar in nature.

Although fairly simple, the three forms of knowledge described above, special purpose, scripts, and rules, have been found adequate for a wide range of application tasks. Much of the early experimental work on this project has been to learn how to use these representations and to try to retrace some of the early robot problem solver work, but with these modern forms of representation.

THE USE OF SENSED KNOWLEDGE VS. STORED KNOWLEDGE

The initial system concept for autonomous vehicles placed heavy reliance on the sensor understanding capabilities of the system to provide the control system with all its knowledge about the local environment, as shown in Figure 1. In that mode of operation the system would essentially be front end limited in capability. The overall performance and capabilities would be limited by the performance of the sensor understanding system. The system would also have a local view of the environment, except for those areas that may have already been explored and stored. The system concept has slowly evolved away from this configuration to one that makes heavy use of stored map knowledge

(e.g. the DMA Database). In this mode, when map knowledge is available, the sensor related information is used to verify and fine tune the map knowledge and provide information about environment dynamics (e.g. moving objects). Thus having map knowledge available not only makes global path planning possible, it also makes the entire problem of vehicle navigation more realistic by relaxing the need for perfect sensor processing.

TESTBED AND PROCESSOR EXPERIMENTS

A very early goal of the project was to provide some capability to demonstrate the system capabilities on a real vehicle to make demonstration and evaluation more effective. It was decided that the first tests would be done on flat terrain (indoor floor), using several movable obstacles and a small vehicle. The control system would be given information about the position of known obstacles and a task to get from one position to another. The system would propose an initial plan for getting to the desired location using the path planner. If the obstacle positions were changed, or if new unknown obstacles were found, however, the script-based planner would then replan the initial plan and attempt to continue to the goal position. The first combination of control software and functioning testbed was successfully demonstrated and video taped in the winter of 1981.

The vehicle used in our first demonstration system was an MIT "Turtle", a primitive DC motor driven vehicle about 9" in diameter by 6" high. A hemispherical shell pivoted from the center interacts with 4

microswitches which act as touch sensors. These sensors are the only feedback element to sense the outside world. The vehicle does not carry its own power source, thus all power and control signals are carried through an umbilical cord. Vehicle control is open loop using timed motor activation to determine both angular and linear displacement, which causes error build-up if extended operating scenarios are attempted.

Any autonomous vehicle control system will contain a number of special purpose processors coupled together loosely through some form of communication network. We are investigating the partitioning of control algorithms and functions to minimize the bandwidth of required communications, which is desirable to increase both the efficiency and reliability of the network links. We have been working with several network concepts to determine the most practical configuration for autonomous vehicle control. Our first implementation used a very primitive vehicle with limited sensory capability to demonstrate a combination of a parallel network for decision making and a hierarchical network for actual vehicle control and sensor processing.

The first parallel network consisted of a mainframe time-share computer (DEC 20) and two Z-80 based microcomputers, as shown in Figure 3. The DEC 20 runs a rule based system to determine the proper course of action for the vehicle. One of the Z-80s runs the specialized problem solver to evaluate optimum paths through a course of known obstacles, and can either find the shortest path or can select a path of maximum concealment. This is one of a number of "problem solving" nodes in the conceptual network. The second Z-80 computer acts as the apex of a two-

computer hierarchical network. This unit recognizes vehicle control messages on the main parallel net, converts them to direction, distance and speed commands, then transmits them to the control computer on board the vehicle. The on-board computer is a single board processor with a built-in Basic interpreter. This computer receives communications from the Z-80 and converts them to proper control signals for driving the vehicle motors.

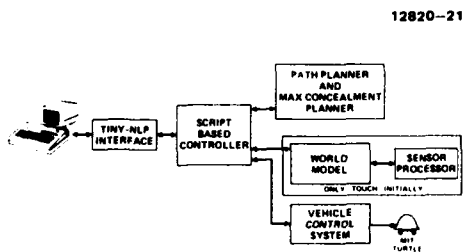


Figure 3.

Touch sensors are also monitored by the on-board processor, and messages are formulated and sent to the Z-80 when obstacles are encountered. The on-board processor also includes enough knowledge to take immediate action based on sensor interpretation. The Z-80 then relays this information to the main network. The reception of sensor data represents an unknown obstacle which is not in the world model accessible to the rule based system. It is necessary for the rule based system to respond to these unknowns, taking appropriate action to replan the vehicle route. This ability to modify behavior based on real-time inputs is fundamental to autonomous vehicle control.

The successful demonstration of our limited "zero-budget" network led to fabrication of an improved vehicle and a much more capable 3-computer vehicle network. The second vehicle, shown in Figure 4, is considerably larger, measuring about 12" x 18" x 12" high. The larger platform provides room to carry a battery pack and two radios which eliminate the need for the umbilical cord. The vehicle is designed to carry a variety of sensors for evaluation of different approaches to the orientation and navigation tasks. Vehicle propulsion is still open loop, using the same techniques (indeed, the same motors) as the MIT Turtle; thus, indirect sensing through the various on-board sensors is relied upon for navigational updates. Currently, the vehicle carries a conventional vidicon camera and two ultrasonic ranging sensors. Touch sensors and infrared ranging sensors are planned for the future.

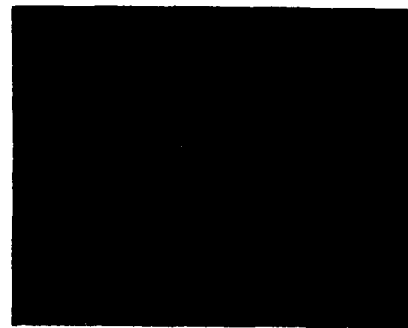


Figure 4.

Our second vehicle network includes our Hierarchical Bus Architecture (HBA) image processing computer (a unique multi-processor system running eighteen 8085 microcomputers in parallel), an advanced Lisp Machine computer built by Symbolics, Inc., and an on-board Z-80 STD Bus computer

for vehicle control. TV camera pointing, Sonar ranging, propulsion, and communication are controlled in real-time by the on-board computer. A wide band one-way RF link from the video camera to the HBA multi-processor system provides real time image processing capability. A moderate bandwidth link between the HBA system and the Lisp Machine, using the Unibus interface, is used to relay processed image data to the Lisp Machine for use in route planning and other higher level decision making tasks. A two-way narrow band RF communication link between the Lisp Machine and the on-board computer completes a triangular network, which is illustrated pictorially in Figure 5.

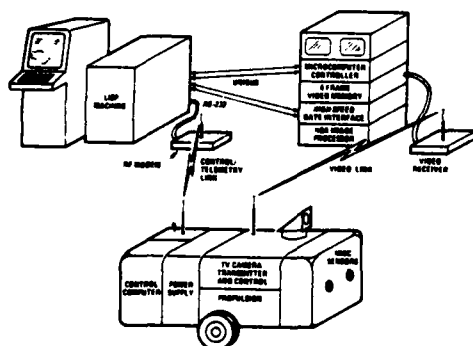


Figure 5.

Our second vehicle, as described above, is primarily a sensor test bed and very little effort has gone into the vehicle's navigation capabilities. We are currently designing a third vehicle which will include an improved on-board navigational capability. This vehicle is based on a larger 18" x 24" tracked platform with a weight carrying capacity of 150 pounds. The use of tracks instead of wheels will provide more capability for negotiating uneven terrain thus

introducing the third dimension into our navigational requirements. To complement this new dimension, a variety of motion sensors is also planned. First, track motion sensors will provide short term measurements for determining vehicle heading and distance covered. Tilt sensors, either simple pendulum sensors or a vertical gyro, will provide information for navigating in the third dimension. A directional gyro will be used for maintaining accurate headings. Capability for multi-processor control of the various vehicle subsystems is also included in the plans for the next generation vehicle.

The network for the third vehicle will include a newer version of our HBA image processing multi-computer. This system, currently under construction, uses multiple Motorola 68000 micro-computers for processing and includes six full-frame memories for data acquisition and intermediate storage. This system is capable of performing low and intermediate level vision processing in near real time, resulting in a list of features extracted from an image. The tentative symbolic features are then relayed to the Lisp machine for symbolic manipulation. The result is a reduction in the processing burden for the Lisp machine thus increasing throughput to allow real-time operation.

The three vehicles we have briefly described here are, of course, simple test beds for algorithm and sensor development. They have been viewed as throw-away vehicles and as such have provided a fairly inexpensive means to gain a great deal of hands-on experience. The ultimate test will come in a real vehicle operating at reasonable speeds in an outdoor environment. Eventually, all of the processing must be carried by the

vehicle to make a truly autonomous system.

CONCLUSION

An early autonomous system capability has been successfully demonstrated. Although the environment and vehicles have been very simple, a great deal has been learned about the required configuration for more capable autonomous systems. The goal of this first generation of system investigation was to retrack the steps of some of the previous projects that used problem-solver based AI methods, but now using knowledge driven methods. This has been successfully accomplished and it has been shown that with even simple knowledge driven control systems, many of the severe limitations of previous attempts can be avoided.

REFERENCES

1. B. L. Bullock, et al, "Image Understanding Application Project: Status Report", proceedings of DARPA Image Understanding Workshop, Stanford University, September 1982.
2. B. Bullock, "The Necessity for a Theory of Specialized Vision," in Computer Vision Systems, E. Riseman (Ed.), Academic Press, 1978.
3. B. Bullock, et.al., Image Understanding Project: Implementation Progress Report, Proc. 1983 IEEE Trends and Applications Conf., National Bureau of Standards, April 1983.
4. B. Bullock, "A General Purpose Perception System Organization That Can Be Tailored to Specialized Applications," Hughes Research Laboratories Research Report, 1983.
5. D. Keirsey, "Natural Language Processing Applied to Navy Tactical Messages," TR 324, NOSC, Feb., 1980.
6. B. L. Bullock, "Unstructured Control Concepts in Scene Analysis," HRL Research Report 497, June 1976 (presented at Proc. 8th Annual Southeastern Symposium on System Theory, University of Tennessee, 1976).
7. B. L. Bullock, "Achieving Performance Flexibility in the Intelligent Bandwidth Compression System," Proceedings of SPIE Electro-Optical Technology for Autonomous Vehicles, Los Angeles, CA, Feb. 1980, Vol. 219.
8. R. Fikes and N. Nilsson, "Strips: A New Approach to Application of Theorem Proving : Problem Solving," IJLAI 71 2.680, 1971.
9. G. G. Hendrix, "Modeling Simultaneous Action and Concurrent Processes," Artificial Intelligence, p. 145-180, Volume 4, 1973.
10. R. Salter, T. Brennan, and D. Friedman, "Concur: A Language for Continuous Concurrent processes," Computer Languages Vol. 5, p. 163-189.

SPATIAL REASONING FOR MOBILITY AND MANIPULATION

Professor Rodney Brooks
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

An autonomous vehicle must plan its path locally to avoid obstacles and it must plan it globally to ensure that it arrives at its desired destination. A system controlling a robot manipulator must plan the trajectory of its payload to avoid collisions while at the same time ensure that the manipulator "elbow" and structural members do not collide with obstacles away from the payload trajectory.

Various representations for space and objects have been developed so that such paths and trajectories can be planned. These include configuration space, variations on grid labelling, and area decomposition and labelling. All have drawbacks. In fact, theoretical work, by others, has proven the existence of algorithms that can always solve such problems, but their running times are high order polynomials and the algorithms do not seem practical. Practical solutions must rely on heuristic approximations and they in turn must rely on an adequate representation of space.

A new representation for space is presented. Space is represented as overlapping free channels. Channel descriptions are readily extracted from either onboard range finders, from a terrain data base, or from geometric models of equipment. As a

vehicle or manipulator link moves, it sweeps out a volume that can be described as a channel. By inverting this computation, given a channel, it is possible to derive constraints on the motion of an object that guarantee that it will stay within the channel.

The channel representation has a number of advantages. The representation of space is not unique and as a consequence requires no normalization of incoming additional spatial information. Thus it is easily updated when information arrives concerning previously uncharted areas or when evidence is gained that contradicts earlier beliefs. The representation is coordinate system independent and leads to an isotropic treatment of space, avoiding the problem of a spatial grid orientation imposing itself on all motions of the vehicle. The basis representation naturally leads to hierarchical representations and the resulting computational space and time savings from hierarchical planning.

This representation demonstrates in the planning of paths for objects over a two dimensional surface, and for trajectory planning for a robot manipulator performing gross motions (as distinct from precision terminal motions) in three-dimensional space.

EXPERT SYSTEMS

WILLIAM B. GEVARTER
NASA Headquarters

ABSTRACT

This paper is an overview of Expert Systems, currently the most popular topic in Artificial Intelligence (AI). Expert Systems are computer programs that use knowledge and reasoning techniques to solve problems difficult enough to normally require the services of a human expert. Topics covered include: what an Expert System is, the structure of Expert Systems, existing systems, constructing an expert system and future trends and applications.

INTRODUCTION

Expert Systems is probably the "hottest" topic in Artificial Intelligence (AI) today. Prior to the last decade, in trying to find solutions to problems, AI researchers tended to rely on nonknowledge-guided search techniques or computational logic. These techniques were successfully used to solve elementary problems or very well structured problems such as games. However, real complex problems are prone to have the characteristics that their search space tends to expand exponentially with the number of parameters involved. For such problems, these older techniques have generally proved to be inadequate and a new approach was needed. This new approach emphasized knowledge rather than search and has led to the field of Knowledge Engineering and Expert Systems. The resultant expert systems technology, limited to academic laboratories in the 70's, is now becoming cost-effective and is

beginning to enter into commercial applications.

WHAT IS AN EXPERT SYSTEM?

Feigenbaum, a pioneer in expert systems, (1982, p. 1) states:

An "expert system" is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field.

The knowledge of an expert system consists of facts and heuristics. The "facts"

constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in a field. The "heuristics" are mostly private, little-discussed rules of good judgement (rules of plausible reasoning, rules of good guessing) that characterize expert-level decision making in the field. The performance level of an expert system is primarily a function of the size and quality of the knowledge base that it possesses.

THE BASIC STRUCTURE OF AN EXPERT SYSTEM

An expert system consists of:

- (1) A knowledge base (or knowledge source) of domain facts and heuristics associated with the problem;
- (2) An inference procedure (or control structure) for utilizing the knowledge base in the solution of the problem;
- (3) A working memory--"global data base"--for keeping track of the problem status, the input data for the particular problem, and the relevant history of what has thus far been done.

A human "domain expert" usually collaborates to help develop the knowledge base. Once the system has been developed, in addition to solving problems, it can also be used to help instruct others in developing their own expertise.

It is desirable, though not yet common, to have a natural language interface to facilitate the use of the system in all three modes: development, problem solving, instruction. In some sophisticated systems, an explanation module is also included, allowing the user to challenge and examine the reasoning process underlying the system's answers. Figure 1-1 is a diagram of an idealized expert system. When the domain knowledge is stored as production rules, the knowledge base is often referred to as the "rule base," and the inference engine as the "rule interpreter."

An expert system differs from more conventional computer programs in several important respects. Duda (1981, p. 242) observes that, in an expert system "...there is a clear separation of general knowledge about the problem (the rules forming a knowledge base) from information about the current problem (the input data) and the methods for applying the general knowledge to the problem (the rule interpreter)." In a conventional computer program, knowledge pertinent to the problem and methods for utilizing this knowledge are all intermixed, so that it is difficult to change the program. In an expert system, "... the program itself is only an interpreter (or general reasoning mechanism) and (ideally) the system can be changed by simply adding or subtracting rules in the knowledge base."

THE KNOWLEDGE BASE

The most popular approach to representing the domain knowledge (both facts and heuristics) needed for an expert system is by production rules (also referred to as "SITUATION-

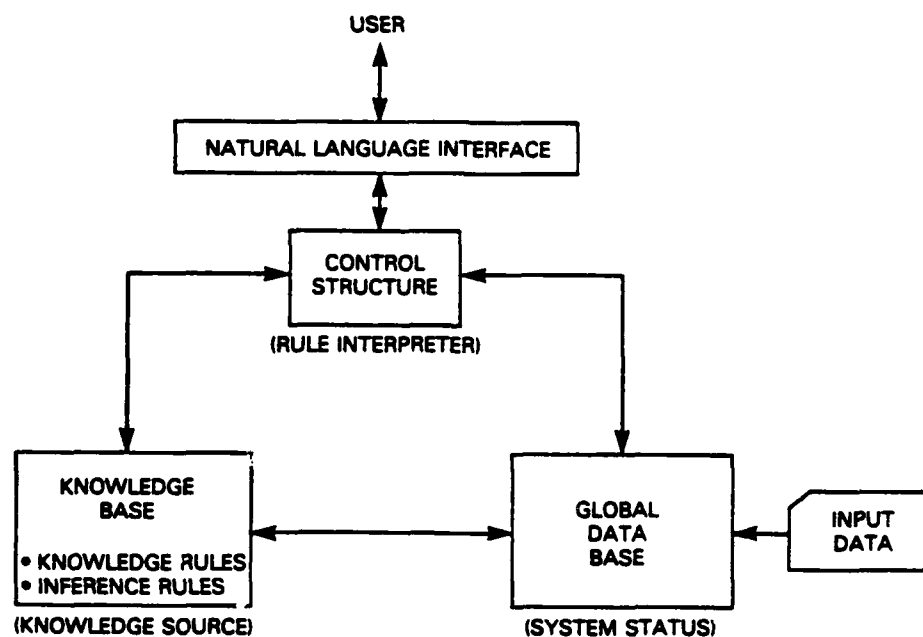


Figure 1-1. Basic Structure of an Expert System.

ACTION rules" or "IF-THEN rules").* Thus, often a knowledge base is made up mostly of rules which are invoked by pattern matching with features of the task environment as they currently appear in the global data base.

THE CONTROL STRUCTURE

In an expert system a problem-solving paradigm must be chosen to organize

* Not all expert systems are rule-based. The network-based expert systems MACSYMA, INTERNIST/CADUCEUS, Digitalis Therapy Advisory, HARPY and PROSPECTOR are examples which are not. Buchanan and Duda (1982) state that the basic requirements in the choice of an expert system knowledge representation scheme are extendibility, simplicity and explicitness. Thus, rule-based systems are particularly attractive.

and control the steps taken to solve the problem. A common, but powerful approach involves the chaining of IF-THEN rules to form a line of reasoning. The rules are actuated by patterns (which, depending on the strategy, match either the IF or the THEN side of the rules) in the global data base. The application of the rule changes the system status and therefore the data base, enabling some rules and disabling others. The rule interpreter uses a control strategy for finding the enabled and for deciding which of the enabled rules to apply. The basic control strategies used may be top-down (goal driven), bottom-up (data driven), or a combination of the two that uses a relaxation-like convergence process to join these opposite lines of reasoning together at some intermediate point to yield a problem solution. However, virtually all the heuristic search and problem solving techniques that the AI community devised have appeared in the various expert systems.

AD-A139 685

PROCEEDINGS OF THE ARMY CONFERENCE ON APPLICATION OF
ARTIFICIAL INTELLIGENCE (U) BATTELLE WASHINGTON
OPERATIONS DC B J TULLINGTON 31 JAN 84

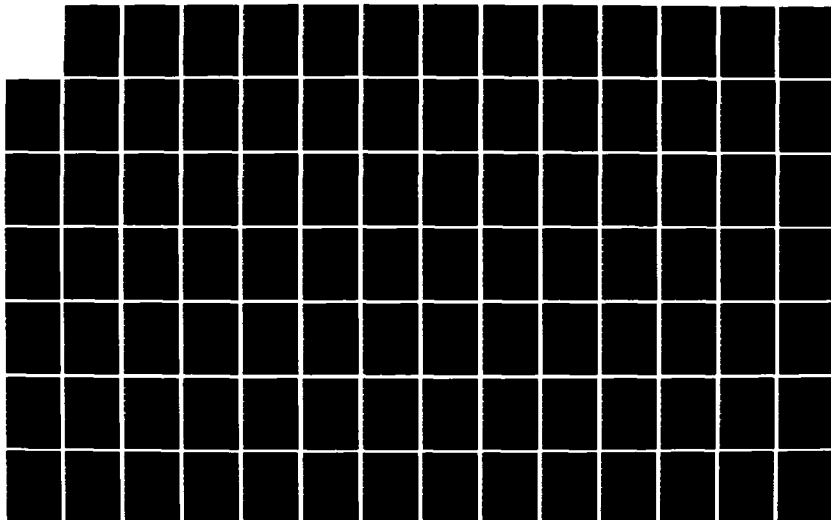
3/

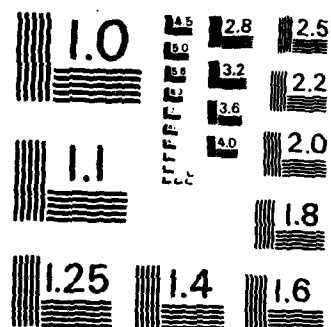
UNCLASSIFIED

DAG29-81-D-0100

F/G 5/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

USES OF EXPERT SYSTEMS

The uses of expert systems are virtually limitless. They can be used to: diagnose, monitor, analyse, interpret, consult, plan, design, instruct, explain, learn and conceptualize.

ARCHITECTURE OF EXPERT SYSTEMS

One way to classify expert systems is by function (e.g., diagnosis, planning, etc.). However, examination of existing expert systems indicates that there is little commonality in detailed system architecture that can be detected from this classification. A more fruitful approach appears to be to look at problem complexity and problem structure and deduce what data and control structures might be appropriate to handle these factors.

The Knowledge Engineering community has evolved a number of techniques (presented in the excellent tutorial by Stefik et al. (1982) and summarized in Gevarter (1982)) which can be utilized in devising suitable expert system architectures.

The use of these techniques in four existing expert systems is illustrated in Tables 1-1-1 through 1-1-4. Tables 1-1-1 through 1-1-4 outline the basic approaches taken by each of these expert systems and shows how the approach translates into key elements of the Knowledge Base, Global Data Base and Control Structure. An indication of the basic control structures of the systems in Tables 1-1-1 through 1-1-4, and some of the other well known expert systems, is given in Table 1-2.

Table 1-2 represents expert system control structures in terms of the search direction, the control techniques utilized, and the search space transformations employed. The approaches used in the various expert systems are different implementations of two basic ideas for overcoming the combinatorial explosion associated with search in real complex problems. These two ideas are:

- (1) Find ways to efficiently search a space,
- (2) Find ways to transform a large search space into smaller manageable chunks that can be searched efficiently.

It will be observed from Table 1-2 that there is little architectural commonality based either on function or domain of expertise. Instead, expert system design may best be considered as an art form, like custom home architecture, in which the chosen design can be implemented from the collection of available AI techniques in heuristic search and problem solving.

EXISTING EXPERT SYSTEMS

Table 1-3 is a list, classified by function and domain of use, of most of the existing major expert systems. It will be observed that there is a predominance of systems in the Medical and Chemistry domains following from the pioneering efforts at Stanford University. From the list, it is also apparent that Stanford University dominates in number of systems, followed by M.I.T., CMU, BBN and SRI, with several dozen scattered efforts elsewhere.

TABLE I-1-1. Characteristics of Example Expert Systems.

SYSTEM: DENDRAL
 INSTITUTION: Stanford University
 AUTHORS: Feigenbaum & Lederberg
 FUNCTION: Data Interpretation

Purpose	Approach	Key Elements of		
		Knowledge Base	Global Data Base	Control Structure
Generate plausible structural representations of organic molecules from mass spectrogram data	1. Derive constraints from the data. 2. Generate candidate structures. 3. Predict mass spectrographs for candidates. 4. Compare with data.	Rules for deriving constraints on molecular structure from experimental data Procedure for generating candidate structures to satisfy constraints Rules for predicting spectrographs from structures	Mass spectrogram data Constraints Candidate structures	Forward chaining Plan, generate and test.

TABLE I-1-2. Characteristics of Example Expert Systems.

SYSTEM: AM
 INSTITUTION: Stanford University
 AUTHORS: Lenat
 FUNCTION: Concept Formation

Purpose	Approach	Key Elements of		
		Knowledge Base	Global Data Base	Control Structure
Discovery of mathematical concepts	Start with elementary ideas in set theory. Search a space of possible conjectures that can be generated from these elementary ideas. Choose the most interesting conjectures and pursue that line of reasoning.	Elementary ideas in finite set theory. Heuristics for generating new mathematical concepts by combining elementary ideas. Heuristics of "interestingness" for discarding bad ideas.	Plausible candidate concepts.	Plan, generate, and test.

TABLE I-1-3. Characteristics of Example Expert Systems.

SYSTEM: RI
INSTITUTION: CMU
AUTHORS: McDermott
FUNCTION: Design

Purpose	Approach	Key Elements of		
		Knowledge Base	Global Data Base	Control Structure
Configure VAX computer systems (from a customer's order of components).	Break problem up into the following ordered subtasks: 1. Correct mistakes in order. 2. Put components into CPU cabinets. 3. Put boxes into unibus cabinets and put components in boxes. 4. Put panels in unibus cabinets. 5. Lay out system on floor. 6. Do the cabling. Solve each subtask and move on to the next one in the fixed order.	Properties of (roughly 500) VAX components. Rules for determining when to move to next subtask based on system state. Rules for carrying out subtasks (to extend partial configuration). (Approximately 1200 rules total)	Customer order. Current task. Partial configuration (System state).	"MATCH" (data driven) (no backtracking)

TABLE I-1-4. Characteristics of Example Expert Systems.

SYSTEM: MYCIN
INSTITUTION: Stanford University
AUTHORS: Shortliffe
FUNCTION: Diagnosis

Purpose	Approach	Key Elements of		
		Knowledge Base	Global Data Base	Control Structure
Diagnosis of bacterial infections and recommendations for antibiotic therapy.	Represent expert judgmental reasoning as condition-conclusion rules together with the expert's "certainty" estimate for each rule. Chain backwards from hypothesized diagnoses to see if the evidence supports it. Exhaustively evaluate all hypotheses. Match treatments to all diagnoses which have high certainty values.	Rules linking patient data to infection hypotheses. Rules for combining certainty factors. Rules for treatment.	Patient history and diagnostic tests. Current hypothesis. Status. Conclusions reached thus far, and rule numbers justifying them.	Backward chaining thru the rules. Exhaustive search.

TABLE I-2. Control Structures of Some Well Known Expert Systems.

System	Function	Domain	Control Structure														
			Search Direction			Control							Search Space Transformations				
			Forward	Backward	Forward and Backward	Exhaustive Search	Generate and Test	Guessing	Relevant Backtracking	Least Commitment	Multilines of Reasoning	Network Editor	Beam Search	Multiple Models	Break into Sub-Problems	Hierarchical Refinement	Hierarchical Resolution
MYCIN	Diagnosis	Medicine	x			x											
DENDRAL	Data Interpr.	Chemistry	x				x										
EL	Analysis	Elec. Circuits	x					x	x								
GUIDON	C.A.I.	Medicine			x												
KAS	Knowl. Acquis.	Geology	x									x					
META-DENDRAL	Learning	Chemistry	x				x										
AM	Concept Formation	Math	x				x										
VM	Monitoring	Medicine			x	x											
GAI	Data Interpr.	Chemistry	x				x										
RI	Design	Computers	x			x									x		
ABSTRIPS	Planning	Robots		x												x	
NOAH	Planning	Robots		x						x					x		
MOLGEN	Design	Genetics			x			x	x	x					x	x	x
SYN	Design	Elec. Circuits	x											x			
HEARSAY II	Signal Interpr.	Speech Unders.		x						x	x						x
HARPY	Signal Interpr.	Speech Unders.	x										x				
CRYSTALIS	Data Interpr.	Crystallography			x		x										x

TABLE I-3. Existing Expert Systems by Function.

Function	Domain	System*	Institution
Diagnosis	Medicine	PIP	M.I.T.
	Medicine	CASNET	Rutgers U.
	Medicine	INTERNIST/CADUCEUS	U. of Pittsburgh
	Medicine	MYCIN	Stanford U.
	Medicine	PUFF	Stanford U.
	Medicine	MDX	Ohio State U.
	Medicine	DART	Stanford U./IBM
	Computer Faults	IDT	DEC
	Computer Faults	REACTOR	E G & G Idaho Inc.
	Nuclear Reactor Accidents		
Data Analysis and Interpretation	Geology	DIPMETER ADVISOR	M.I.T./Schlumberger
	Chemistry	DENDRAL	Stanford U.
	Chemistry	GAI	Stanford U.
	Geology	PROSPECTOR	SRI
	Protein Crystallography	CRYNALIS	Stanford U.
	Determination of Causal Relationships in Medicine	RX	Stanford U.
	Determination of Causal Relationships in Medicine	ABEL	M.I.T.
	Oil Well Logs	ELAS	AMOCO
	Electrical Circuits	EL	M.I.T.
	Symbolic Mathematics	MACSYMA	M.I.T.
Analysis	Mechanics Problems	MECHO	Edinburgh
	Naval Task Force Threat Analysis	TECH	Rand/NOSC
	Earthquake Damage Assessment for Structures	SPERIL	Purdue U.
	Digital Circuits	CRITTER	Rutgers U.
	Computer System Configurations	RI/XCON	C.M.U./DEC
	Circuit Synthesis	SYN	M.I.T.
	Chemical Synthesis	SYNCHEM	SUNY Stonybrook
Design			

*References to these systems can be found in Duda (1981), Siefik, et al. (1982), Buchanan (1981), Buchanan and Duda (1982), Barr and Feigenbaum (1982), IJCAI-81, and AAAI-82.

TABLE I-3. Existing Expert Systems by Function. (cont.)

Function	Domain	System*	Institution
Planning	Chemical Synthesis	SECHS	U. of Cal. Santa Cruz
	Robotics	NOAH	SRI
	Robotics	ABSTRIPS	SRI
	Planetary Flybys	DEVISER	JPL
	Errand Planning	OP-PLANNER	Rand
	Molecular Genetics	MOLGEN	Stanford U.
	Mission Planning	KNOBS	MITRE
	Job Shop Scheduling	ISIS-II	CMU
	Design of Molecular Genetics Experiments	SPEX	Stanford U.
	Medical Diagnosis	HODGKINS	M.I.T.
Learning from Experience	Naval Aircraft Ops	AIRPLAN	CMU
	Tactical Targeting	TATR	RAND
	Chemistry	METADENDRAL	Stanford U.
	Heuristics	EURISKO	Stanford U.
	Mathematics	AM	CMU
	Speech Understanding	HEARSAY II	CMU
	Speech Understanding	HARPY	CMU
	Machine Acoustics	SU/X	Stanford U.
	Ocean Surveillance	HASP	System Controls Inc.
	Sensors On Board Naval Vessels	STAMMER-2	NOSC, San Diego/SDC
Concept Formation	Medicine—Left Ventrical Performance	ALVEN	U. of Toronto
	Military Situation Determination	ANALYST	MITRE
	Patient Respiration	VM	Stanford U.
	Structural Analysis	SACON	Stanford U.
	Computer Program		
	Electronic Troubleshooting	SOPHIE	B.B.N.
	Medical Diagnosis	GUIDON	Stanford U.
	Mathematics	EXCHECK	Stanford U.
	Steam Propulsion Plant Operation	STEAMER	BBN
	Diagnostic Skills	BUGGY	BBN
Monitoring	Causes of Rainfall	WHY	BBN
	Coaching of a Game	WEST	BBN
	Coaching of a Game	WUMPUS	M.I.T.
	Coaching of a Game	SCHOLAR	BBN
Use Advisor			
Computer Aided Instruction			

TABLE I-3. Existing Expert Systems by Function. (cont.)

Function	Domain	System*	Institution
Knowledge Acquisition	Medical Diagnosis	TEIRESIAS	Stanford U.
	Medical Consultation	EXPERT	Rutgers
Expert System Construction	Geology	KAS	SRI
		ROSIE	Rand
		AGE	Stanford U.
		HEARSAY III	USC/ISI
		EMYCIN	Stanford U.
		OPS 5	CMU
		RAINBOW	IBM
		KMS	U. of MD
		EXPERT	Rutgers
		ARBY	Smart Sys. &
		MECS-AI	Tokyo U.
		BATTLE	NRL AI Lab
Consultation/Intelligent Assistant	Battlefield Weapons Assignments	Digitalis Therapy Advisor	M.I.T.
	Medicine	RAYDEX	Rutgers U.
	Radiology	XCEL	CMU/DEC
	Computer Sales	ONCOCIN	Stanford U.
	Medical Treatment	CSA Model-Based Nuclear Power Plant Consultant	GA Tech
Management	Nuclear Power Plants	RECONSIDER	U. of CA, S.F.
	Diagnostic Prompting in Medicine		
Automatic Programming	Automated Factory Project Management	IMS	CMU
	Modelling of Oil Well Logs	CALLISTO	DEC
Image Understanding		ΦNIX	Schlumberger-Doll Res.
		CHI	Kestrel Inst.
		PECOS	Stanford U.
		LIBRA	Stanford U.
		SAFE	VSC/ISI
		DEDALUS	SRI
		Programmer's Apprentice	M.I.T.
		VISIONS	U. of Mass.
		ACRONYN	Stanford U.

CONSTRUCTING AN EXPERT SYSTEM

Duda (1981, p. 262) states that to construct a successful expert system, the following prerequisites must be met:

- There must be at least one human expert acknowledged to perform the task well.
- The primary source of the expert's exceptional performance must be special knowledge, judgment, and experience.
- The expert must be able to explain the special knowledge and experience and the methods used to apply them to particular problems.
- The task must have a well-bounded domain of application.

Using present techniques and programming tools, the effort required to develop an expert system appears to be converging towards five man-years, with most endeavors employing two to five people in the construction.

SUMMARY OF THE STATE-OF-THE-ART

Buchanan (1981, pp. 6-7) indicates that the current state of the art in expert systems is characterized by:

- Narrow domain of expertise. Because of the difficulty in building and maintaining a large knowledge base, the typical domain of expertise is narrow. The principal exception is INTERNIST, for which the knowledge base covers 500 disease diagnoses. However, this

broad coverage is achieved by using a relatively shallow set of relationships between diseases and associated symptoms. (INTERNIST is now being replaced by CADUCEUS, which uses causal relationships to help diagnose simultaneous unrelated diseases.)

- Limited knowledge representation languages for facts and relations.
- Relatively inflexible and stylized input-output languages.
- Stylized and limited explanations by the systems.
- Laborious construction. At present, it requires a knowledge engineer to work with a human expert to laboriously extract and structure the information to build the knowledge base. However, once the basic system has been built, in a few cases it has been possible to write knowledge acquisition systems to help extend the knowledge base by direct interaction with a human expert, without the aid of a knowledge engineer.
- Single expert as a "knowledge czar." We are currently limited in our ability to maintain consistency among overlapping items in the knowledge base. Therefore, though it is desirable for several experts to contribute, one expert must maintain control to insure the quality of the data base.
- Fragile behavior. In addition, most systems exhibit fragile behavior at the boundaries of their capabilities. Thus, even some of the best systems

come up with wrong answers for problems just outside their domain of coverage. Even within their domain, systems can be misled by complex or unusual cases, or for cases for which they do not yet have the needed knowledge or for which even the human experts have difficulty.

- Requires knowledge engineer to operate. Another limitation is that for most current systems only their builders or other knowledge engineers can successfully operate them—a friendly interface not having yet been constructed.

Nevertheless, Randy Davis (1982) observes that there have been notable successes. A methodology has been developed for explicating informal knowledge. Representing and using empirical associations, five systems have been routinely solving difficult problems--DENDRAL, MACSYMA, MOLGEN, RI and PUFF--and are in regular use. The first three all have serious users who are only loosely coupled to the system designers. DENDRAL, which analyzes chemical instrument data to determine the underlying molecular structure, has been the most widely used program (see Lindsay et al., 1980). RI, which is used to configure VAX computer systems, has been reported to be saving DEC twenty million dollars per year, and is now being followed up with XCON. In addition, as indicated in Table I-3, dozens of systems have been constructed and are being experimented with.

FUTURE TRENDS

Figure I-2 lists some of the expert systems applications currently under development.

It will be observed that there appear to be few domain or functional limitations in the ultimate use of expert systems. However, the nature of expert systems is changing. The limitations of rule-based systems are becoming apparent. Not all knowledge can be readily structured in the form of empirical associations. Empirical associations tend to hide causal relations (present only implicitly in such associations). Empirical associations are also inappropriate for highlighting structure and function.

Thus, the newer expert systems are adding deep knowledge having to do with causality and structure. These systems will be less fragile, thereby holding the promise of yielding correct answers often enough to be considered for use in autonomous systems, not just as intelligent assistants.

The other change is a trend towards an increasing number of non-rule based systems. These systems, utilizing semantic networks, frames and other knowledge representations, are often better suited for causal modeling and representing structure. They also tend to simplify the reasoning required by providing knowledge representations more appropriate for the specific problem domain.

Figure I-3 (based largely on Hayes-Roth IJCAI-81 Expert System tutorial and on Feigenbaum, 1982) indicates some of the future opportunities for expert systems. Again, no limitation is apparent.

It thus appears that expert systems will eventually find use in most endeavors which require symbolic reasoning with detailed professional knowledge--which includes much of the world's work. In the process, there will be exposure and refinement of the

- Medical diagnosis and prescription
- Medical knowledge automation
- Chemical data interpretation
- Chemical and biological synthesis
- Mineral and oil exploration
- Planning/scheduling
- Signal interpretation
- Signal fusion situation interpretation from multiple sensors
- Military threat assessment
- Tactical targeting
- Space defense
- Air traffic control
- Circuit diagnosis
- VLSI design
- Equipment fault diagnosis
- Computer configuration selection
- Speech understanding
- Intelligent Computer-Aided Instruction
- Automatic Programming
- Intelligent knowledge base access and management
- Tools for building expert systems

Figure 1-2. Expert System Applications Now Under Development.

- *Building and Construction*
Design, planning, scheduling, control
- *Equipment*
Design, monitoring, control, diagnosis, maintenance, repair, instruction.
- *Command and Control*
Intelligence analysis, planning, targeting, communication
- *Weapon Systems*
Target identification, adaptive control, electronic warfare
- *Professions*
(Medicine, law, accounting, management, real estate, financial, engineering)
Consulting, instruction, analysis
- *Education*
Instruction, testing, diagnosis, concept formation and new knowledge development from experience.
- *Imagery*
Photo interpretation, mapping, geographic problem-solving.
- *Software*
Instruction, specification, design, production, verification, maintenance
- *Home Entertainment and Advice-giving*
Intelligent games, investment and finances, purchasing, shopping, intelligent information retrieval
- *Intelligent Agents*
To assist in the use of computer-based systems
- *Office Automation*
Intelligent systems
- *Process Control*
Factory and plant automation
- *Exploration*
Space, prospecting, etc.

Figure 1-3. Future Opportunities for Expert Systems.

previously private knowledge in the various fields of applications.

On a more near-term scale, in the next few years we can expect to see expert systems with thousands of rules. In addition to the increasing number of rule-based systems we can also expect to see an increasing number of non-rule-based systems. We can also expect much improved explanation systems that can explain why an expert system did what it did and what things are of importance.

By the late 80's, we can expect to see intelligent, friendly and robust human interfaces and much better system building tools.

Somewhere around the year 2000, we can expect to see the beginnings of systems which semi-autonomously develop knowledge bases from text. The result of these developments may very well herald a maturing information society where expert systems put experts at everyone's disposal. In the process, production and information costs should greatly diminish, opening up major new opportunities for societal betterment.

REFERENCES

1. Barr, A., and Feigenbaum, E. A., The Handbook of Artificial Intelligence, Vol. 2, Los Altos, CA: W. Kaufman, 1982.
2. Buchanan, B. G., "Research on Expert Systems," Stanford University Computer Science Department, Report No. STAN-CS-81-837, 1981.
3. Buchanan, B. G. and Duda, R. O., "Principles of Rule-Based Expert Systems," Heuristic Programming Project Report No. HPP 82-14, Department of Computer Science, Stanford, CA, Aug. 1982. (To appear in Advances in Computers, Vol. 22, M. Yorit. (ed.), New York: Academic Press)
4. Davis, R., "Expert Systems: Where Are We: and Where Do We Go From Here?" AI Magazine, Vol. 3. No. 2, Spring 82, pp. 3-25.
5. Duda, R. O. "Knowledge-Based Expert Systems Come of Age," Byte, Vol. 6, No. 9, September 1981, pp. 238-281.
6. Feigenbaum, E. A., "Knowledge Engineering for the 1980's," Computer Science Department, Stanford University, 1982.
7. Gevarter, W. B., An Overview of Expert Systems, NBSIR 82-2505, National Bureau of Standards, Washington, D.C., May 1982 (Revised October 1982).
8. Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A., and Lederberg, J., Applications of Artificial Intelligence for Organic Chemistry: The ENDRAL Project, New York: McGraw-Hill, 1980.
9. Stefik, M., Alkins, J., Balzer, R., Benoit, J., Birnbaum, L. Hayes-Roth, R., Sacerdoti, E., "The Organization of Expert Systems, A Tutorial, Artificial Intelligence, Vol. 18, 1982, pp. 135-173.
10. IJCAI-81. The International Joint Conference on AI, Vancouver, August, 1981.
11. AAAI-82. Proc. of the National Conference on AI, CMU & U. of Pittsburgh, Pittsburgh, PA, August 18-22, 1982.

AD P003035

KNOWLEDGE ACQUISITION AND EVALUATION WITHIN EXPERT SYSTEMS

D. W. Loveland
Duke University

ABSTRACT

Knowledge acquisition and evaluation are essential to maintaining the expertise of expert systems. ~~Here we summarize some of the major efforts to date in knowledge acquisition and indicate work that we are undertaking in the just emerging but crucial area of knowledge evaluation.~~ *Here we summarize some of the major efforts to date in knowledge acquisition and indicate work that we are undertaking in the just emerging but crucial area of knowledge evaluation.*

INTRODUCTION

Expert systems have gathered much publicity lately as a few of the systems actually move out of the laboratory and into the real world. The existence of such systems has removed the field of artificial intelligence (AI) from the completely esoteric with the realization that AI technology is beginning to affect our approach to real problems.

Informed observers realize that building expert systems is still an art and applicable to quite constrained domains. Not only are the domains constrained but these systems initially capture (a portion of) one expert's knowledge for one time slice, the period over which the system is being

built. Either the problem is a static one or the system must be updated. Thus the problem of updating a knowledge base can be as significant as the first definition of that base.

The problem of collecting the knowledge for the original system and updating it afterward is called **knowledge acquisition**. A major subtask within the knowledge acquisition task is that of **knowledge evaluation**, the task of determining the value of the knowledge base. By this we mean insuring that the knowledge base was correct information and that the knowledge chunks (e.g., rules) interact appropriately with one another. This is what logicians call **soundness** of a system; consistency is not entirely the correct notion here. A system can be consistent without being correct, and consisting can be costly to insure. Of course, a sound system is consistent, but one might have local consistency adequate to the demands of the users without full global consistency. The methods we discuss here regarding knowledge evaluation will emphasize soundness rather than consistency.

The research reported here undertaken by the author has received partial support from the Air Force Office of Sponsored Research, Grant AFOSR 81-0221.

The purpose of this paper is to present quickly a sense of the state-of-the-art in knowledge acquisition, with special attention to knowledge evaluation. This is done by reference to a few of the better defined examples, chosen also to highlight different approaches. Most of our presentation concerns rule-based systems, although the reader should be aware that semantic nets and frames are inference architectures also used in expert systems.

APPROACHES TO THE PROBLEM

Knowledge occurs at three levels:

- 1) Service aids. This includes editing systems that permit information to be entered and manipulated at the text modification level. Somewhat more convenient are formatting aids that present stencils or skeletons that allow the user to "fill in blanks." Along with these devices often come support packages like spelling correctors that prevent mundane syntactic errors from disrupting the session. We do not consider this category further, although such tools are important aids.
- 2) Intelligent assistants. Systems exist that give substantial aid to the acquisition of knowledge. Some commercial systems to aid acquisition are coming into existence but the more sophisticated systems are still highly experimental. One of the most sophisticated systems is TEIRESIAS, developed by Davis (3), that serves as the acquisition system for MYCIN, a rule-based expert system for infectious disease (see (9)). TEIRESIAS aids both rule acquisition and atomic component acquisition by building models of the type and content of the knowledge already present. This allows the system to have an expectation of the knowledge being presented. We consider TEIRESIAS again later in the paper.
- 3) Automated rule acquisition. An alternative method of knowledge acquisition is automated learning (inductive inference) schemes, an approach that has already demonstrated its capability in certain settings. We will comment further on two rule acquisition systems that incorporate learning, Meta-DENDRAL (see (1)) and PLANT (see (7)). Other interesting systems exist, such as EXPERT (see (1)).

Knowledge evaluation is an emerging area of concern and thus less exploration has occurred in this area. Two complementary areas of concern are receiving attention.

- 1) Local evaluation. One-step inferences are being evaluated for soundness, consistency, completeness and redundancies. Completeness here means that it is possible to ask if all cases of a particular vector of attributes are considered, a matter easily checked at the one-step inference level. An example of this approach to knowledge evaluation is given by Suwa, Scott and Shortliffe in their work with ONCOCIN (see (10)). Related work has been done by Politakis, Weiss and Kulikowsky regarding the EXPERT system (see (8)).

- 2) Global evaluation. The user is ultimately concerned with the functioning of the entire inference system, and it is well-known that if inference chains are of any length at all (even length three or so) then isolated rules can look correct but the interaction can be incorrect. Present-day knowledge bases generally have several hundred rules, and as new rules are added, the likelihood of unintended interactions is substantial. We illustrate this later. Work on global knowledge evaluation is being pursued by Politakis, Weiss and Kulikowsky (see (8)) and by Loveland and Valtorta (see (6)). The former group has made use of standard testing of recorded cases to maintain a sound knowledge base. The latter investigators are exploring special test procedures tuned to key properties that should not be violated.

EXAMPLE SYSTEMS AND PROCEDURES

We will illustrate some of the comments made earlier by examples from past or ongoing investigations. Space prohibits fuller discussions, but interested readers are urged to consult the articles referenced for more details.

TEIRESIAS best represents (at present) the intelligent assistant approach to rule acquisition. A typical rule, here written for the task of car selection rather than infection-diagnosis, appears as follows.

If

- (1) The income of person is upper-middle,

- (2) the sex of person is male,
(3) the age of person is middle-aged,
(4) the person has interest in automobiles

then

there is evidence (0.7) that the person would buy a sport-sedan.

Corresponding to the exterior form of the rule, which appears as just given, there is an internal form in which (almost) every line has the form

(predicate function, object, attribute, value)

For example, the first premise would have representation (SAME, PERSON, INCOME, UPPER-MIDDLE) where the object PERSON would be replaced by an object-variable named CONTEXT and SAME denotes 'is'. Since the inference system reasons backwards, rules are categorized by their conclusion (called ACTION) part, and within the conclusion by the attribute. TEIRESIAS aids rule acquisition by building rule models which collect frequency counts on attributes of premises of all rules in the same category, both by straight occurrence and correlations with other attributes within the premise. When a new rule of the same category is entered by the expert, TEIRESIAS has some expectation as to the incoming text and can use these expectations to help translate the English input and also catch inadvertently omitted conditions in the new rule. Although the rule base must be sizeable already for useful rule models to be built, and the rules must be similar enough within the category to have the frequency measure useful, when these conditions do exist the added contribution of this semantic information is significant. TEIRESIAS also provides structures

that inform about structures associated with the various primitive terms, such as the attributes and values. This allows the expert to add a new attribute, for example, although this application domain expert may not have much familiarity with the actual program. For more information on TEIRESIAS see (3), (4).

Meta-DENDRAL, in contrast, is an example of rule acquisition by automated learning. The underlying program, Heuristic DENDRAL, has the task of determining likely molecular structures, the topology or geometry of the molecules, given nontopological information about the molecules, such as atomic components and, principally, the mass spectrometer data for the molecules. The latter gives fracture information about the molecule when bombarded with electrons. (The DENDRAL project is perhaps the oldest, most famous and best developed of the projects now identified with the label of "expert system".)

DENDRAL uses a "plan-generate-test" organization where, under simple constraints all possible bondings are generated and then the fragments that electron bombardment would yield are determined. This output is compared to the actual mass spectrometer results and a ranking of plausible structures is then printed out.

The information regarding fragmentation (i.e., simulation of the mass spectrometer) is retained in rule form, which allows for easy correction and augmentation. As the domain of molecules to be analyzed increased and new domains were tried, it became clear that automated rule acquisition would be beneficial (and also an interesting test-bed for new ideas and techniques in AI). Meta-DENDRAL was developed to create rules

that simulate the behavior of the mass spectrometer. It is an inductive inference problem organized in the same manner as DENDRAL itself, by "plan-generate-test". Input to Meta-DENDRAL is a set of input-output pairs (I/O pairs) for a mass spectrometer; precisely, one molecular structure is coupled with a spectral line that represents one fragment obtainable from electron bombardment, and the quantity thereof. Output from Meta-DENDRAL is a collection of new rules that add to the simulation of the mass spectrometer within DENDRAL. The procedure is roughly as follows. For each molecular structure input, a set of possible fragmentations was noted. The allowed fragmentations were governed by a flexible and naive "half-order theory" that prevented bond breaks at unlikely places (e.g., no double bond breaks). This kept the number of fragmentations feasible. A rough estimate of the value of each candidate is determined by the number of I/O pairs with a spectral line (e.g., fragment) of one of the fragments of the candidate. This score accompanies the candidate. Then patterns that constitute the premises of the new rules are sought, i.e., subgraphs that start with the most general pattern and acquire features until no input I/O pair that satisfies the subgraph condition yields a false fragment for the rule conclusion. After this step, further pruning of rules and generalizing conditions using more specialized laws than the "half-order theory" are employed before the new rules are output.

The automated learning of rules has proven to be quite successful with some rules found that were unknown to chemists; some have been published in the chemical literature. Worth noting are the points where specific chemical information is used,

observing that this knowledge is not very sophisticated. This inductive inference is effective only when a large sample of I/O pairs is available. This is a common property of inductive inference schemes.

For more information regarding Meta-DENDRAL see (2). For a thorough treatment of the entire DENDRAL project see (5).

PLANT provides another example in automated learning. It is particularly interesting because Michalski and Chilausky implemented a rough "controlled experiment" where the same expert system used rule acquisition via input from experts and also via inductive inference. In performance the rule system acquired inductively performed somewhat better than the experts' rule system. Even though this was a highly constrained environment the experiment is impressive since the domain is "real", not a toy. Again, a large number of samples are needed to permit the learning mechanisms to converge. For information on this experiment see (7).

KNOWLEDGE EVALUATION

We conclude with brief illustrations of approaches to knowledge evaluation.

The local evaluation approach is illustrated by Suwa, Scott and Shortliffe where for each attribute a table is built (in effect a decision table). In the table is recorded the premises of each rule that concludes a (possible range of) value(s) for that attribute. Where overlaps in the premises occur, consistency of

outcome is checked. Redundant or subsumed rules are spotted as are premise conditions not covered. When applied to the ONCOCIN System this approach appears to yield dividends in maintaining a useful rule base. For more information see (10).

The global evaluation approach has two aspects at present. Politakis, Weiss and Kulikowsky have employed known test cases to insure that, whatever the rules look like locally, the relation between the input and output of the system (the "bottom line") is performing as desired, at least on the known test cases. For more information see (8).

Loveland and Valtorta are beginning to explore approaches to global evaluation intended for expert systems over a domain too extensive to check entirely by known test cases. The approach to be mentioned here is akin to program verification in spirit. Important properties of large classes of expert systems are identified and methods designed to inform the user when these properties are violated, or at least possibly violated. In the example to be considered the product is an improved test procedure for detecting violation of the designated property. Rather than random testing one is given a methodology for reducing strongly the number of test cases needed to catch a possible violation. The property examined is classification ambiguity, applicable to diagnostic expert systems where many "meaningful" inputs are expected to yield a unique classification. Although the method now known only works for conventional true-false logic systems, not inexact systems (a severe restriction), the extension to inexact systems seems likely and is under study. We remark that the extension appears sensitive to the basic definition of acceptance.

The technique of defining the test procedure for finding classification ambiguity involves regarding the inference system as a graph (or net), first finding the input nodes of influence for each output classification, and, if the potential ambiguity cannot be resolved at this point, proceeding with a (slightly nonstandard) testing procedure based on the "divide-and-conquer" idea. One seeks to locate "minimally ambiguous" input vectors, of which the meaningful but ambiguous vectors are usually a subset. These minimally ambiguous vectors are displayed to the expert for him to decide if indeed they represent unintended ambiguities. For more on this technique see (6).

We conclude with an example that illustrates that local evaluation in itself is insufficient, and where a test case might not be provided because of the unexpected output. The example falls within the domain of diagnosis ambiguity. The problem domain is automobile repair. The rule to be added is:

If
 the fuel-gauge has power-off,
 then
 the fuel-gauge reads empty.

This rule is likely to be approved by a local check. Suppose we run a global ambiguity check as just outlined above. We could get the following display:

The input set of symptoms.

- 1) engine stalls
- 2) all electrical devices fail

The output diagnosis

- D1) some electrical wire is disconnected,
- D2) run out of gas.

Although D1 is reasonable, D2 would catch most people by surprise and be deemed erroneous. But observe how it could arise via reasonable rules. A trace of the deduction might reveal a rule as follows:

(*) If
 1) engine stalls
 Then
 gas-tank is empty.

From the fact "gas-tank is empty" the conclusion that one has run out of gas is an immediate conclusion. The rule (*) is certainly reasonable in local context. However, now we can deduce that: all electrical devices fail, so the fuel-gauge power is off, so (by the new rule) the fuel-gauge reads empty; so the gas-tank is empty. Thus we get our surprising result.

The problem, of course, is that rules with what was once an adequate set of conditions may need other conditions added when a new rule is entered. (The rule (*) now needs a condition that not all the power is off.) This is quite common situation. Thus we argue that global evaluation techniques of some sophistication (i.e., other than fixed test cases) will be needed.

REFERENCES

1. Buchanan, B. G., and E. A. Feigenbaum. Dendral and Meta-Dendral: Their applications dimension. *Artif. Intell.* 11, 1978, 5-24. Also in *Readings in Artif. Intell.* (Webber and Nilsson, eds.), Tioga Press, Palo Alto, 1981, 313-320.
2. Buchanan, B. G., G. L. Sutherland, and E. A. Feigenbaum. Heuristic DENDRAL: A program for generating explanatory hypotheses in organic chemistry. *Mach. Intell.* 4 (Meltzer and Michie, eds.), Edinburgh Univ. Press, Edinburgh, 1969.
3. Davis, R. TEIRESIAS: Applications of meta-level knowledge. *Knowledge-based Systems in Artif. Intell.* (Davis and Lenat), McGraw-Hill, New York, 1982.
4. Davis, R. Interactive transfer of expertise: Acquisition of new inference rules. *Artif. Intell.* 12, 1979, 121-157. Also in *Readings in Artif. Intell.* (Webber and Nilsson, eds.), Tioga Press, Palo Alto, 1981, 410-428.
5. Lindsay, R., B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. Applications of Artificial Intelligence for Organic Chemistry: the DENDRAL Project. McGraw-Hill, New York, 1980.
6. Loveland, D. W., and M. Valtorta. Detecting ambiguity: An example of knowledge evaluation. *Proc. Eighth Intern. Joint Conf. on Artif. Intell.*, Karlsruhe, W. Germany, August 1983, 182-184.
7. Michalski, R. S., and R. L. Chilausky. 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. *Intern. Jour. of Policy Anal. and Info. Systems*, 1980, 125-161.
8. Politakis, P., S. Weiss, and C. Kulikowski. Designing consistent knowledge bases for EXPERT consultation systems. *Rutgers Computer Sci. Report CBM-TR-100*, 1979.
9. Shortliffe, E. H. *Computer-Based Medical Consultations: MYCIN*. Elsevier, New York, 1976.
10. Suwa, M., A. C. Scott, E. H. Shortliffe. An approach to verifying completeness and consistency in a rule-based expert system. *Stanford Computer Sci. Report STAN-CS-82-922*, 1982.
11. Weiss, S., and D. Kulikowski. EXPERT: A system for developing consultation models. *Proc. Sixth Intern. Joint Conf. on Artif. Intell.*, 1979, 942-947.

APPLICATIONS OF KNOWLEDGE ENGINEERING

H. Penny Nii
Stanford University

Knowledge engineering is the art of designing and building computer programs that perform specific tasks by combining methods of symbolic reasoning with knowledge of the task domain. The resulting programs are often called knowledge-based, or Expert, programs. The discovery and cumulation of techniques of symbolic inference and representation is generally the work of the field of artificial intelligence research. The knowledge of the problem domain is held by experts of the domain. This knowledge consists of both formal, textbook knowledge and experiential knowledge--the "expertise" of the experts. The task of integrating AI methods with knowledge of the problem domain to build Expert Systems falls to the knowledge engineers.

For a program to qualify as an Expert System, it needs more than the domain knowledge within the program. The following characteristics, taken together, define Expert Systems:

- a clear separation of knowledge base and inference methods;
- the ability to capture and express both knowledge and line-of-reasoning in forms directly understandable by human experts in the application domain;
- ability of the system to explain its line-of-reasoning; and
- a knowledge base that is incrementally refined, from the

start of system development and throughout its life.

Several Expert System architectures are currently in wide use. One of the simplest of these is the goal-oriented, backward chaining of inference rules, an architecture used in the MYCIN and PROSPECTOR programs. Knowledge in these systems are represented as IF-THEN rules; the control is an exhaustive search; and uncertain data and knowledge are handled by certainty factor calculations. Numerous Expert Systems have been developed using this architecture, and many of them have been developed using EMYCIN, a software tool for developing MYCIN-like Expert Systems.

One of the most complex architectures in wide use is the Blackboard Model, first developed for the HEARSAY-II speech understanding program. Since then, this architecture has been used successfully in signal interpretation, information fusion, planning and image understanding problems. Unlike MYCIN-like systems, knowledge can be represented as frames, rules, and/or procedures. The control is an opportunistic island-driving, whereby many and diverse knowledge sources are brought to bear as opportunities for their use arise in the problem solving process. Two attempts have been made to create a tool for developing Blackboard programs--AGE and HEARSAY-III. Of the two, AGE is generally available and has been used to develop several Expert Systems.

EXPERIMENTAL LOGIC
and
THE AUTOMATIC ANALYSIS OF ALGORITHMS

Dr. Frank M. Brown
Dept. Computer Sciences
The University of Texas at Austin

INTRODUCTION

Experimental Logic can be viewed as a branch of logic dealing with the actual construction of useful deductive systems and their application to various scientific disciplines. In a sense it is a reversion of the study of logic back to its original purpose, before the study of logic became merely the metamathematical study of artificial language systems.

In this paper we describe an experimental logic called Quantified Computational Logic (i.e., QCL) and an automatic theorem prover called the SYMBolic EVALuator (i.e., SYMEVAL) which automatically makes deductions in the QCL language. This logic is being applied to solving problems in several areas of computer science such as automatic complexity analysis of computer programs, automatic verification of the correctness of computer programs, automatic natural language analysis, and as a model for advanced language design.

SYMEVAL AND QUANTIFIED COMPUTATIONAL LOGIC

Quantified Computational Logic is an experimental theorem proving and programming language interpreted by the SYMBolic EVALuator automatic theorem prover. This theorem prover is based on the following fundamental principle about deduction:

The Fundamental Deduction Principle. Almost all steps in an automatic deduction should be viewed and usually take place by a method of replacement of expressions by equivalent expressions, and that the smaller such expressions are, the better. A few important steps may involve generalization of the theorem, but these are deliberate steps taken with due consideration--not the mindless application of basic generalizing inference rules.

We have come to believe this principle for two basic reasons:

1. The first reason is our overall impression of the many automatic theorem provers we have personally constructed in the last decade, that the closer we adhere to the above principle the better is the resulting automatic theorem prover because of the lack of redundancy in representing the same information throughout the proof process.

2. The second reason is that theorem provers have an important role to play in inductive reasoning, and that this role takes the form of using the theorem prover to simplify complex expressions containing unknown free variables into equivalent sentences which explicitly state the values of those variables. Note that if the resulting sentence was not equivalent because of a generalization step during the course of the deduction, then even if the resulting sentence was a solution to the unknown, there would still be no assurance that that solution was the only solution. Thus, a generalizing step would not produce all the needed inductive information.

SOME SIMPLE CONSEQUENCES

The consequences of this principle are startling, and frankly, at first we were loathe to accept them because they seem to contradict much of the past and current research on automatic theorem proving (including our own). For example, almost all past research on automatic theorem proving including both resolution systems and sequent calculus (i.e., natural deduction) systems, has been based on the Prawitz-Robinson Unification algorithm (Prawitz, Robinson) which does not generally satisfy our principle. Consider the example where we have an axiom $(F X 2)=(G X)$, and a theorem $(\text{IMPLIES}(F I Y)(P Y))$. Then by unifying $(F X 2)$ with $(F I Y)$, binding X to I and Y to 2 , and replacing $(F I 2)$ by $(G I)$ we can deduce the expression: $(\text{IMPLIES}(G I)(P 2))$. But this expression is not generally equivalent in the theory to the original theorem because, although $(F I 2)=(G I 2)$, $(\text{IMPLIES}(F I 2)(P 2))$ is only an instance of $(\text{IMPLIES}(F I Y)(P Y))$.

A second consequence is that Robinson's general resolution (Robinson) inference rule (even ignoring the previously mentioned unification problem) does not generally satisfy this principle. For example, in the case of propositional logic, the resolution inference rule says that $(C \text{ OR } D)$ may be inferred from $(L \text{ OR } C)$ and $((\text{NOT } L) \text{ OR } D)$. Since $(C \text{ OR } D)$ is not equivalent to either of the above expressions or their conjunction, it cannot generally satisfy this principle. (Note, however, that there are certain cases of the resolution rule which do satisfy this principle. One such case is the in unit resolution of propositional logic where L or rather $L=T$ is an axiom of our theory, and $((\text{NOT } L) \text{ OR } D)$ is the expression being simplified. In this case, the resolvent is D , which is equivalent to the original expression being simplified. Unit resolution, sometimes called "forward chaining" or "backward chaining" in natural deduction theorem provers, has been found to be especially useful in certain theories (Bledsoe 1,2, Pastre).

From these example consequences, one can well imagine the difficulty of constructing a useful deductive system satisfying this principle. However, after much perseverance, we have constructed a useful (but incomplete) deductive system for QCL which satisfies this principle. This deductive system is the SYMMETRIC LOGIC whose rules are described later.

This principle also has consequences with respect to the richness of the logical language used by a theorem prover. In particular, richer logical languages, such as those which include unrestricted quantifiers, seem to have an advantage in being able to express axioms in accordance with this principle. One good example of this is the ELIM rule of the SYMMETRIC LOGIC which is stated schematically:

$$\begin{aligned} &(\text{EQUAL}(h \ X \ Z1...Zn) \\ &\quad (\text{EX } l1...lm \ (\text{AND}(\text{EQUAL}(\text{const } Z1...Zn \ l1...lm)X) \\ &\quad \quad (c \ Z1...Zn \ l1...lm)))) \end{aligned}$$

The closest one can write axioms of this form in a quantifier free language such as (Boyer1) seems to be:

$$\begin{aligned} &(\text{IMPLIES}(h \ X \ Z1...Zn) \\ &\quad (\text{AND}(\text{EQUAL}(\text{const } Z1...Zn(d1 \ X \ Z1...Zn)...dm \ X \ Z1...Zn))X) \\ &\quad \quad (c \ Z1...Zn(d1 \ X \ Z1...Zn)...dm \ X \ Z1...Zn))) \end{aligned}$$

which can be obtained from ELIM by replacing the existential quantifiers (EX li(p li)) by the skolem function selectors (di Z1...Zn) and reducing the EQUAL to an IMPLY. The problem with the latter axiom scheme is that its use may cause a generalizing step to be made in the course of the proof. Another problem is that the introduction of the extra (di Z1...Zn) expressions necessitates the use of a later generalization to get rid of them, thus, again causing another generalization. Finally, because the:

$$(c \ Z1...Zn(d1 \ X \ Z1...Zn)...dm \ X \ Z1...Zn))$$

is logically implied by: (c Z1...Zn l1...lm), it follows that it might be true without (c Z1...Zn l1...lm) being true. In such a case, the needed generalization expression c would have to be explicitly given to the system.

CONSEQUENCES FOR ADVANCED PROGRAMMING LANGUAGE DESIGN. We believe that our fundamental deduction principle applies also to deducing purely computational theorems such as those which are deduced when an automatic theorem prover is used as an interpreter of programs written in a logic such as QCL. Because linear Horn Clause Resolution theorem provers such as (Kowalski, Chester, Colmerauer), and SL Resolution theorem provers such as MOORE's Baroque (Moore) contradict this principle both with respect to unification and the resolution rule, it follows that such systems will not produce all the possible answers to a given problem.

LINGUISTIC CONVENTIONS OF SYMEVAL

The language of the SYMBolic EVALuator consists of constant symbols and variable symbols. There are two types of constant symbols: FUNSYMs and ATOMSYMs. There are also two types of variable symbols: variables and SCHEMATORS. Constant symbols are analogous to nouns and verbs in a natural language, whereas variable symbols are analogous to pronouns.

VARIABLES. A variable symbol is any literal atom LITATOM which is not an ATOMSYM not occurring after a left parenthesis. For example, X is not a variable, but Y is a variable in (X Y).

SCHEMATOR. A schemator is any list of the form: (schemator.symbol arg1...argN) whose schemator symbol is on the SCHEMATOR list. The initial schemator symbols are SCH1 SCH2 SCH3 SCH4.

$$\begin{aligned} &(\text{EQUAL}(h \ X \ Z1...Zn) \\ &\quad (\text{EX } l1...lm \ (\text{AND}(\text{EQUAL}(\text{const } Z1...Zn \ l1...lm)X) \\ &\quad \quad (c \ Z1...Zn \ l1...lm)))) \end{aligned}$$

The closest one can write axioms of this form in a quantifier free language such as (Boyer1) seems to be:

$$\begin{aligned} &(\text{IMPLIES}(h \ X \ Z1...Zn) \\ &\quad (\text{AND}(\text{EQUAL}(\text{const } Z1...Zn(d1 \ X \ Z1...Zn)...dm \ X \ Z1...Zn))X) \\ &\quad \quad (c \ Z1...Zn(d1 \ X \ Z1...Zn)...dm \ X \ Z1...Zn)))) \end{aligned}$$

which can be obtained from ELIM by replacing the existential quantifiers (EX li(p li)) by the skolem function selectors (di Z1...Zn) and reducing the EQUAL to an IMPLY. The problem with the latter axiom scheme is that its use may cause a generalizing step to be made in the course of the proof. Another problem is that the introduction of the extra (di Z1...Zn) expressions necessitates the use of a later generalization to get rid of them, thus, again causing another generalization. Finally, because the:

$$(c \ Z1...Zn(d1 \ X \ Z1...Zn)...dm \ X \ Z1...Zn))$$

is logically implied by: (c Z1...Zn l1...lm), it follows that it might be true without (c Z1...Zn l1...lm) being true. In such a case, the needed generalization expression c would have to be explicitly given to the system.

CONSEQUENCES FOR ADVANCED PROGRAMMING LANGUAGE DESIGN. We believe that our fundamental deduction principle applies also to deducing purely computational theorems such as those which are deduced when an automatic theorem prover is used as an interpreter of programs written in a logic such as QCL. Because linear Horn Clause Resolution theorem provers such as (Kowalski, Chester, Colmerauer), and SL Resolution theorem provers such as MOORE's Baroque (Moore) contradict this principle both with respect to unification and the resolution rule, it follows that such systems will not produce all the possible answers to a given problem.

LINGUISTIC CONVENTIONS OF SYMEVAL

The language of the SYMBolic EVALuator consists of constant symbols and variable symbols. There are two types of constant symbols: FUNSYMs and ATOMSYMs. There are also two types of variable symbols: variables and SCHEMATORS. Constant symbols are analogous to nouns and verbs in a natural language, whereas variable symbols are analogous to pronouns.

VARIABLES. A variable symbol is any literal atom LITATOM which is not an ATOMSYM not occurring after a left parenthesis. For example, X is not a variable, but Y is a variable in (X Y).

SCHEMATOR. A schemator is any list of the form: (schemator.symbol arg1...argN) whose schemator symbol is on the SCHEMATOR list. The initial schemator symbols are SCH1 SCH2 SCH3 SCH4.

Every constant symbol is either a function symbol or an atom symbol. Some function symbols are quantifiers or modal symbols. The list of primitive, defined, and declared symbols of each type can be obtained by typing the following global variables:

FUNSYM. List of all currently defined functional symbols. A FUNSYM symbol f applied to N arguments is written as $(f \text{ arg1} \dots \text{argN})$. We call this a function value of f . The number of arguments may be zero in which case the function is written (f) . Some example functions in QCL are: IF, EQUAL, ALL, EX, LAMBDA, APPLY, QUOTE, QCL.

ATOMSYM. List of all currently defined atom symbols. An atom symbol x is always written as x without parentheses. Some example ATOMS YMS in QCL are: NIL, T.

QUANTSYM. List of all currently defined quantifier symbols. All quantifiers are of the form: $(q \ v \ (f \ v) \ x_1 \dots x_n)$. Quantifiers must contain their associated bound variable in their first argument position. This variable is bound only in the first and second argument positions. Additional quantifiers may be defined, declared, or axiomatized. All such additional quantifiers should be added to the QUANTSYM list: $(\text{SETQ QUANTSYM}(\text{CONS new.quantifier QUANTSYM}))$. Some example primitive quantifiers in QCL are: ALL, EX, LAMBDA.

MODALSYM. List of all currently defined modal symbols. An example modalsym in QCL is: QCL.

SYMEVAL

The SYMBolic EVALuator is a general interpreter of the Frege's quantificational logic (Frege) (i.e., first order logic with schemators, but not higher order logic). Higher order logic is handled by axiomatizing it within first order logic. The actual symbols and inference rules of the logic are arbitrary as far as SYMEVAL is concerned. SYMEVAL evaluates a function value $(f \text{ arg1} \dots \text{argN})$ in one of two ways. If the function f is not an FSUBRSYM, then it applies the definitions, axioms, and rules about f to the result of evaluating each argument. However, if f is an FSUBRSYM, then only the first argument is evaluated before the f laws are applied. SYMEVAL's symbolic evaluation of expressions is thus somewhat analogous to LISP's (McCarthy) method of evaluating expression with the exception that SYMEVAL returns a "normalform" expression equal to the input expression whereas LISP returns the meaning of that "normalform" expression. Thus, whereas $(\text{CONS}(\text{QUOTE A})(\text{QUOTE B}))$ evaluates to its meaning: $(A.B)$ in LISP, it SYMBolically EVALuates to the equal expression $(\text{QUOTE}(A.B))$. The evaluation to an equal expression allows SYMEVAL to handle quantified variables in a graceful manner. For example, whereas the evaluation of (CONS X Y) with unbound variables X and Y gives an error in LISP, its SYMBolic EVALuation results in the equal expression (CONS X Y) .

FSUBRSYM. List of functions to be evaluated as FSUBRSYMS. For example, QCL has the primitive FSUBRSYMS: IF and the defined FSUBRSYMS: AND, OR, LOR, IMPLIES, IMPLY.

Specifically, SYMEVAL works as follows given as input an expression E to evaluate, an association list A of bindings to apply, an association list H of hypotheses, a list FL of recursive functions being tentatively unraveled, a list of universal variables which may be solved for, and a list QE of existential variables which may be solved for.

1. If E is a VARIABLE or a SCHEMATOR, then if E is bound in the association list A, then the result of replacing E by its binding in A is returned, and if it is not bound in A, then E itself is returned.
2. Otherwise, if E is an explicitly quoted name (e.g., (QUOTE FOO)) or an automatically quoted name such as a number or string, then E itself is returned.
3. Otherwise, if E is an ATOMIC SYMBOL, then the result of APPLYing axioms to that atomic symbol is returned.
4. Otherwise, if E is a QUANTifier SYMBOL, then it is of the form: (quantifier.symbol bound.var bound.arg unbound.arg1 ... unbound.argN). In this case, the expressions in the unbound argument positions are recursively SYMBOLically EVALuated using the input values of A, H, FL, QA, and QE. The expression in the bound argument position is recursively SYMBOLically EVALuated with A and H changed to the result of deleting any bindings containing the bound variable, with the type of the bound variable also added to H, with FL unchanged, with QA set to NIL unless the quantifier symbol is ALL, in which case the bound variable is added to QA, and with QE set to NIL unless the quantifier symbol is EX, in which case the bound variable is added to QE. Finally, the axioms about that quantifier are SYMBOLically APPLYed to the expression resulting from evaluating the arguments as described.
5. Otherwise, if E is an FSUBR SYMBOL, then only the first argument of E is SYMBOLically EVALuated with QA and QE changed to NIL. The other arguments are merely replaced by the result of substituting any free variables or schemators in E by their bindings in A. The axioms about that FSUBR SYMBOL are then SYMBOLically APPLYed to the expression resulting from evaluating and substituting the arguments as described.
6. Otherwise, E is of the form (function.symbol arg1 ... argN) and every argument of E is recursively SYMBOLically EVALuated without changing A, H, FL, QA, or QE. The axioms about the symbol are then SYMBOLically APPLYed to the expression resulting from evaluating the arguments as described.

The SYMBOLic EVALuator function calls the SYMBOLic EVALuator APPLY function in order to apply the axioms about a given symbol to an expression starting with that symbol. Specifically, SYMEVAPPLY works as follows given as input an expression E to apply, an association list H of hypotheses, a list FL of recursive functions being tentatively unraveled, a list of universal variables which may be solved for, and a list QE of existential variables which may be solved for.

1. If E is an explicitly quoted or automatically quoted name, then E is returned.
2. Otherwise, if E has the form (function.symbol arg1 ... argN), the function symbol is not the FUNSYM list or if E has an atomic symbol not on the ATOMSYM list, then E itself is returned.
3. Otherwise, if E has the form (function.symbol arg1 ... argN), the function symbol is the name of a function defined in INTERLISP and the arguments are all explicitly or automatically quoted objects, then the result of KWOTEing the INTERLISP evaluation of E is returned.

4. Otherwise, if the type of E is NIL then NIL is returned, and if the type of E is T then T is returned.
5. Otherwise, E is an ATOMIC SYMBOL, or begins with a FUNCTION SYMBOL. If that symbol has an explicit definition, or a recursive definition such that some subset of the arguments in E form a known measured subset for that recursive definition, then the result of recursively SYMBOLICALLY EVALUATING the result of using that definition is returned.
6. Otherwise, if any axioms or rules about this symbol can be used, then the result of recursively SYMBOLICALLY EVALUATING the result of using the first such usable axiom or rule is applied.
7. Otherwise, the FNORMALFORM axiom is used, if it can be used.
8. Otherwise, if the symbol has a recursive definition and that function symbol is not already on the list of function symbols FL being tentatively unraveled, then it is put on that list, and the result of SYMBOLICALLY EVALUATING the result of using that definition is compared with E itself, and the more useful of the two expressions is returned.
9. Otherwise, E itself is returned.

All axioms, including axioms which axiomatize the logical symbols such as AND, OR, NOT, ALL, and EX, are essentially of the form:

(IMPLIES c(EQUAL p q))

In accordance with the Fundamental Deduction Principle, such an axiom is used to replace an expression p by an expression q which is logically equal to p given that c is true in the given theory and context. For example, because the c expression of a definition is the true symbol T, a definition is used to simply replace p by q generally.

Thus, given an ATOMIC SYMBOL E, or a list E consisting of a FUNCTION.SYMBOL followed by zero or more arguments, SYMEVAPPLY tries to use axioms whose p expression either is or begins with that symbol as follows:

1. If SYMEVAPPLY is trying to use a definition, then the p expression of that definition is MATCHED to the expression E. The result returned is the result of using the bindings obtained from the MATCHING to substitute for the free occurrences of variables and schemators in the q expression of the definition.
2. Otherwise, if SYMEVAPPLY is trying to use an axiom, then the p expression of that axiom is MATCHED to the expression E. If no match is possible, then this axiom is not usable on E and that fact is returned and the next axiom is tried. If a MATCH is possible, then the bindings obtained from this MATCH are used to substitute for the free variables and schemators in both the c and q expressions of this axiom. If the result of repeatedly recursively SYMBOLICALLY EVALUATING the new c expression is not a non-NIL value, then again the axiom is not usable and the next one is tried. If the result is a non-NIL value, then the axiom is usable and the new q expression is returned.

Although all axioms are essentially of the form:

(IMPLIES c(EQUAL p q))

and even though the use of schemators allows numerous kinds of axiom schemes to be represented in QCL, there are still a number of useful schemes which cannot be represented directly within the QCL language without ascending to the meta level. For this reason, SYMEVAL allows such schemes to be represented as INTERLISP functions. SYMEVAL will attempt to use any such axiom scheme represented as an INTERLISP function in the same manner as it uses any other axiom. The interface to the INTERLISP function is this: The arguments to the function are the current E, H, FL, QA, and QE. The result returned by the function is either the instantiated q expression of the hypothetical usable axiom represented by the function, or E itself, if the hypothetical axiom was unusable. In any case, the prime requirement is that the expression returned from such a function must be logically equivalent in the given theory and in the context H to the input expression E. In accordance with INTERLISP's argument conventions, the function need not have formals for H, FL, QA, and QE; in which case, they are ignored. Such functions may obtain their result in any manner they wish, including calling the theorem prover itself.

FUNCTION OBJECTIFICATION

Function symbols of a theory can be theoretically divided into two classes. Namely, the symbols of the theory which appear essentially as arguments to the p expression in an axiom of the form: $c \Rightarrow p = q$ and the symbols which do not. For example, the CONS function is clearly an object because of the SHELL axiom: $(CAR(CONS X Y)) = X$. However, the recursive function APPEND is not an object, at least until we prove some theorems about it, because there is no axiom of the form: $(f(APPEND X Y)) = q$. It is important to determine which non-primitive symbols are objectified at a given point in an extensible theory (Brown11) in order to avoid their evaluation in order to get the effect of a call by need evaluation (Brown4,6).

PRIMITIVE SYMBOLS OF QCL

Quantified Computational Logic consists of both primitive symbols and non-primitive symbols. The primitive symbols of QCL listed below are axiomatized by the ADDRULE command described later.

NIL	"falsity" "normal end of a list"
T	"truth"
(IF p l r)	"if p then l else r" FSUBRSYM
(EQUAL x y)	"x equals y", or if x and y are sentences "x if and only if y"
(ALL v x)	"for all v x" QUANTSYM

(EX v x)	"for some v x" QUANTSYM
(QUOTE x)	treats x as a name
(LAMBDA v x)	"the function mapping v to x" QUANTSYM
(AP f a)	"the application of function f to a"
(LAMBDAF f)	"f is a good function"
(BADLAMBDAF f)	"f is a bad function"
(QCL x)	"x is QCL-true" MODALSYM

TYPES

Every symbol in QCL has one or more types associated with its function value. In addition, the bound variable of each quantifier is also of one or more types. The initial types are: NIL, T, LAMBDAF, BADLAMBDAF, OTHERN. The NIL type consists of the atom NIL. The T type consists of the atom T. The LAMBDAF type consists of those functions which can be arguments to an application. The BADLAMBDAF type are the other functions. The OTHERN type consists of those numbers which are not recursively constructed by the shell principle. An expression is of type NIL or T iff it is EQUAL to that value: (EQUAL X NIL) or (EQUAL X T). For any other type, an expression is recognized to be that type by a recognizer symbol in QCL whose name is that type. For example (LAMBDAF X) iff X is of type LAMBDAF, and (BADLAMBDAF X) iff X is of type BADLAMBDAF. Also types for lists, positive integers, negative integers, decimal numbers, strings, and literal atoms are initially created types produced by the SHELL principle described below. These types are called: LISTP, PNUMBERP, NNUMBERP, DECIMALP, STRINGP, LITATOMP. NIL and T are not LITATOMPs because they have their own type.

UNIVERSE. The list of all currently defined types in the system. LITATOMP STRINGP DECIMALP NNUMBERP PNUMBERP LISTP NIL T LAMBDAF BADLAMBDAF OTHERN OTHER .

(SETQ XXX(PRINTTYPES)) (PP XXX). Pretty prints the type of the function value of every symbol in the system.

The type of the function value of a primitive symbol is already known. The type of the bound variable of a primitive quantifier is already known. The type of the function value of a newly defined symbol is automatically created when that symbol is defined. However, the type of the bound variable of a newly defined quantifier must be explicitly declared by the following command:

(VASSUME quantifier.symbol types).

For example, a newly defined universal quantifier ALL.LISTS.AND.STRINGS ranging over only lists and strings would need the command:

(VASSUME 'ALL.LISTS.AND.STRINGS '(LISTP STRINGP)).

THE SYMMETRIC LOGIC DEDUCTION SYSTEM

INTRODUCTION

The SYMEVAL theorem prover now runs with a new logical system called SYMMETRIC LOGIC which treats universal and existential quantifiers in an analogous manner. For example, as suggested by (Wang2) several years ago, SYMMETRIC LOGIC rewrites both of the following sentences to (FOO A), after evaluating their subexpressions, when trying to prove them:

(ALL X (IMPLIES (EQUAL X A) (FOO X)))

(EX X (AND (EQUAL X A) (FOO X)))

Thus, the essence of the SYMMETRIC LOGIC technique is to push quantifiers to the lowest scope possible in hopes of finding a way to eliminate them. Thus, unlike the sequent calculus (Szabo,Brown1,3,4,6,12,25) and other logic systems (Bledsoel,2,Bibel2) based on the Prawitz-Robinson Unification algorithm (Robinson), which essentially loses the scope of the existential quantifier during the skolemization process, SYMMETRIC LOGIC handles equalities very well indeed. The power of a logic which handles equalities like this is very convincing, in an application domain dealing essentially with equations such as real algebra, logic programming, and language analysis. SYMMETRIC LOGIC is the synthesis of several earlier logic systems including the initial symmetric logic used by the real algebra rule package (Brown24), and the bind logic used by the logic programming and natural language rule packages (Brown26). Current research is aimed mainly at synthesizing the SYMMETRIC LOGIC method of handling quantifiers with the method used by our sequent calculus system into one general system. The purpose of synthesizing these different logical systems is to eventually develop one simple, systematic, yet general, logic system capable of performing well in many application domains.

PROPOSITIONAL LOGIC

The syntax of SYMMETRIC LOGIC includes three propositional symbols:

NIL meaning: false

T meaning: true

(IF p l r) meaning: if p then l else r FSUBR

IF treats all non-Boolean objects (e.g., LISTPs, NUMBERPs, and OTHERS) as if they were T. An expression x is Boolean iff x is T or NIL when evaluated in some world. Thus any non-NIL object is assumed to be a true value. For example: (IF 44 l 2) is l. The NIL and T atom symbols are SUBRs, whereas the IF symbol is FSUBR. Thus, only the first argument of IF is SYMbolically EVALuated before the IF axioms are applied.

The SYMIF rule is used on an expression (IF p l r) in the following manner:

1. First the type of p is determined. If it is NIL, then r is returned, and if it does not include NIL, then l is returned. These rules can be written schematically as: 1. If the type of p does not contain NIL, then (IF p l r) = l. 2. If p=NIL, then (IF p l r) = r.
2. Otherwise, if l is of type NIL and r is not, r is of type NIL and l is not, the l-effects of p contribute significantly to binding the free variables in l, or the r-effects of p contribute significantly to binding the free variables of r, then the IFNORMALFORM scheme is applied to (IF p l r). The effects of p consist of the equality statements in p of the form (EQUAL v t) or (EQUAL t v), hereafter called bindings.
3. If certain conditions hold, then IFNORMALFORM.

If neither of the above cases hold, then p is assumed to be true with the H list set accordingly, p is solved for a variable if possible and the A list is set accordingly (i.e., that solution is added as an additional binding and is also used to modify any other bindings), and l is SYMBolically EVALuated. Next, p is assumed to be false and the H list is set accordingly and r is SYMBolically EVALuated.

SYMIF then tries to use the following rules in the given order:

4. If l=r then (IF p l r) = r
5. If l differs from r only in bindings contained in the context list A then (IF p l r) = r
6. If p => l=r then (IF p l r) = r
7. If p=l, r=NIL then (IF p l r) = p
8. If p is Bool, l=T, r=NIL then (IF x l r) = x
9. IFNORMALFORM

This rule is not the scheme:

$$(IF (IF p a b) l r) = (IF p (IF a l r) (IF b l r))$$

which is far too inefficient for effective deduction. Instead the following more general scheme is used:

$$(IF (IF p \dots a \dots) L R) = (IF p \dots (IF a l r) \dots)$$

for every largest subexpression not containing an IF symbol. Furthermore, if a=NIL, then (IF a l r) becomes r, and if the type of a does not include r, then (IF a l r) becomes l.

10. Equality Substitution

This rule applies the scheme:

$$(IF(EQUAL x y) (p x y)) = (IF(EQUAL x y) (p x l y l))$$

after solving the equation (equal x y) for an interesting subexpression u.

IF applies the IFNORMALFORM axiom before evaluating l and r whenever:

- (1) l and r occur no more than once in the resulting expression (i.e., if the truth values are known, we get:

$$(IF (IF p NIL NIL) l r) = \text{by above axioms: } (IF NIL l r) = r$$

$$(IF (IF p NIL Tor) l r) = (IF p r l)$$

$$(IF (IF p Tor NIL) l r) = (IF p l r)$$

$$(IF (IF p Tor Tor) l r) = (IF p l l) = l$$

Tor means T or any non-boolean exp.)

- (2) The bindings in x contribute significantly to the evaluation of l or they contribute significantly to the evaluation of r.
- (3) The type restrictions in x contribute significantly to l or to 4 (not yet implemented).

The SYMMETRIC propositional logic also includes the functional normal form rule which eliminates embedded occurrences of the IF symbol. The FNORMALFORM axiom embodies:

$$(f x l \dots x_n (IF p l r) y l \dots y_m) = (IF p (f x l \dots x_n l y l \dots y_m) (f x l \dots x_n r y l \dots y_m))$$

EQUALITY LOGIC

The syntax of SYMMETRIC LOGIC includes an equality symbol:

(EQUAL l r) meaning: l equals r

The EQUAL symbol is a SUBR. Thus, all arguments to EQUAL are SYMbolically EVALuated before the EQUAL axioms are applied. Generally speaking, equality has four fundamental properties:

1. Equality is reflexive: $x=x$
2. Equality is symmetric: $x=y$ implies $y=x$

3. Equality is transitive: $x=y$ implies $(y=z$ implies $x=z)$
4. Equality is extentional: $x=y$ implies $(Px$ implies $Py)$
 $x=y$ implies $(y=z) = (x=z)$
 $x=y$ implies $fx = fy$

The EQUAL axioms are given below.

The first two equality laws deal with the reflexivity of equality and the inequality of distinct data objects.

1. $(EQUAL\ X\ X) = T$
2. if l and r are different data objects, then $(EQUAL\ l\ r) = NIL$

The next six axioms reduce EQUAL to IF whenever possible.

3. $(EQUAL\ NIL\ r) = (IF\ r\ NIL\ T)$
4. $(EQUAL\ l\ NIL) = (IF\ l\ NIL\ T)$
5. if l is Boolean in a world, then $(EQUAL\ l\ T) = l$
6. if r is Boolean in a world, then $(EQUAL\ T\ r) = r$
7. $(EQUAL\ l\ (EQUAL\ x\ y)) = (IF\ (EQUAL\ x\ y)\ (EQUAL\ l\ T)\ (EQUAL\ l\ NIL))$
8. $(EQUAL\ (EQUAL\ x\ y)\ r) = (IF\ (EQUAL\ x\ y)\ (EQUAL\ r\ T)\ (EQUAL\ r\ NIL))$

This law reduces the equality of functions (i.e., Frege's Werthverlauf) to the equality of their function values on equal arguments.

9. $(EQUAL\ (LAMBDA\ x(p\ x))\ (LAMBDA\ y(p\ y)))$
 $= (ALL\ V(EQUAL\ (APPLY\ (LAMBDA\ x(p\ x))\ V)\ (APPLY\ (LAMBDA\ y(p\ y))\ V)))$

QUANTIFICATIONAL LOGIC

SYMMETRIC LOGIC also has two quantifiers:

$(ALL\ v\ p)$ meaning: for all v, p

$(EX\ v\ p)$ meaning: for some v, p

The ALL axioms are:

0. $ALL\ t$ type p is not NIL $\implies (ALL\ v\ p) = T$
1. $ALL\ \sim v$ $(ALL\ v\ p) = p$
2. $ALL\ v v$ $(ALL\ v\ v) = NIL$

- 3.ALL= $(\text{ALL } v((\text{not}(\text{EQUAL } v \text{ t}))\dots\text{or}(\text{p } v))) = (\text{p } t)$
- 4.ALL recog $(\text{ALL } v(\text{recog } v)) = \text{NIL}$
- 5.QUANTIF $(\text{ALL } v(\text{IF } p \text{ (I } x)(r \text{ x}))) = (\text{IF } p (\text{ALL } v(\text{I } v)) (\text{ALL } v(r \text{ v})))$
- 6.ALLand $(\text{ALL } v(\dots\text{or}(\text{and } (x \text{ v}) (y \text{ v}))\dots))$
 $(\text{and}(\text{ALL } v(\dots\text{or}(\text{x v})\dots))(\text{ALL } v(\dots\text{or}(\text{y v})\dots)))$
- 7.ALLor
- 8.ALLnot $(\text{ALL } v(\text{IF } p \text{ NIL } T)) = (\text{IF } (\text{EX } v \text{ p}) \text{ NIL } T)$
- 9.ALLident $(\text{ALL } v(\text{IF } p \text{ T NIL})) = (\text{IF } (\text{ALL } v \text{ p}) \text{ T NIL})$
- 10.ELIM

The EX axioms are:

- 0.EX† type p is not NIL $\Rightarrow (\text{EX } v \text{ p}) = \text{T}$
- 1.EX~v $(\text{EX } v \text{ p}) = \text{p}$
- 2.EXvv $(\text{EX } v \text{ v}) = \text{T}$
- 3.EX= $(\text{EX } v((\text{EQUAL } v \text{ t})\dots\text{and}(\text{p } v))) = (\text{p } t)$
- 4.EXrecog $(\text{EX } v(\text{recog } v)) = \text{T}$
- 5.QUANTIF $(\text{EX } v(\text{IF } p \text{ (I } x)(r \text{ x}))) = (\text{IF } p (\text{EX } v(\text{I } v)) (\text{EX } v(r \text{ v})))$
- 6.EXor $(\text{EX } v(\dots\text{and}(\text{or } (x \text{ v}) (y \text{ v}))\dots))$
 $= (\text{or}(\text{EX } v(\dots\text{and}(\text{x v})\dots))(\text{EX } v(\dots\text{and}(\text{y v})\dots)))$
- 7.EXand
- 8.EXnot $(\text{EX } v(\text{IF } p \text{ NIL } T)) = (\text{IF } (\text{ALL } v \text{ p}) \text{ NIL } T)$
- 9.EXident $(\text{EX } v(\text{IF } p \text{ T NIL})) = (\text{IF } (\text{EX } v \text{ p}) \text{ T NIL})$
- 10.ELIM

EX-OR does:

1. $(\text{IF } a (\text{IF } p \text{ I } r) \text{ NIL}) = (\text{IF } p (\text{IF } a \text{ I NIL}) (\text{IF } a \text{ r NIL}))$
2. if I is Boolean then $(\text{IF } p \text{ I } r) = (\text{IF } (\text{IF } p \text{ I NIL}) \text{ T } (\text{IF } p \text{ NIL } r))$
3. $(\text{EX } v(\text{IF } (a \text{ v}) \text{ T } (b \text{ v}))) = (\text{IF } (\text{EX } v(a \text{ x})) (\text{EX } v(b \text{ v})) \text{ T})$

ALL-AND does:

1. $(\text{IF } a (\text{IF } p \mid r) T) = (\text{IF } p (\text{IF } a \mid T) (\text{IF } a \mid r) T)$
2. if $\mid$ is Boolean then $(\text{IF } p \mid r) = (\text{IF } (\text{IF } p \mid T) (\text{IF } p \mid r) \text{NIL})$
3. $(\text{ALL } v(\text{IF}(a \mid v)(b \mid v)\text{NIL})) = (\text{IF}(\text{ALL } v(a \mid x))(\text{ALL } v(b \mid v))\text{NIL})$

The Elimination axioms have the forms:

$(\text{EQUAL } (h \mid X \mid Z_1 \dots Z_n)$
 $(\text{EX } \mid_1 \dots \mid_m (\text{AND}(\text{EQUAL } (\text{const } Z_1 \dots Z_n \mid_1 \dots \mid_m) X)$
 $(c \mid Z_1 \dots Z_n \mid_1 \dots \mid_m))))$

They have the effect of the rewrite schemas:

$(\text{EQUAL } (\text{ALL } X (s \mid X))$
 $(\text{AND}(\text{ALL } Z_1 \dots (\text{ALL } Z_n (\text{ALL } \mid_1 \dots (\text{ALL } \mid_n$
 $(\text{IMPLIES } (c \mid Z_1 \dots Z_n \mid_1 \dots \mid_m)$
 $(s(\text{const } Z_1 \dots Z_n \mid_1 \dots \mid_m)))))))$
 $(\text{ALL } Z_1 \dots (\text{ALL } Z_n (\text{ALL } X (\text{IMPLIES}(\text{NOT}(h \mid X \mid Z_1 \dots Z_n)$
 $(s \mid X))))))))$

$(\text{EQUAL } (\text{EX } X (s \mid X))$
 $(\text{OR } (\text{EX } Z_1 \dots (\text{EX } Z_n (\text{EX } \mid_1 \dots (\text{EX } \mid_m$
 $(\text{AND}(c \mid Z_1 \dots Z_n \mid_1 \dots \mid_m)$
 $(s(\text{const } Z_1 \dots Z_n \mid_1 \dots \mid_m)))))))$
 $(\text{EX } Z_1 \dots (\text{EX } Z_n (\text{EX } X (\text{AND}(\text{NOT}(h \mid Z_1 \dots Z_n)$
 $(s \mid X))))))))$

ABSTRACTION LOGIC

The SYMMETRIC LOGIC has a primitive symbol, LAMBDA, for functional abstraction (i.e., for forming the werthverlauf of a function value) and a primitive symbol, AP, for function application.

$(\text{LAMBDA } v(p \mid v))$
 $(\text{AP } g \mid a)$

Five axioms and theorems similar to the set theory abstraction axioms used in (Brown^{4,6}) are assumed for lambda conversion:

- T1. $(\text{BADLAMBDA } p \mid X) ==> (\text{AP } S \mid X) = \text{NIL}$
- T2. $(\text{NOT}(\text{BADLAMBDA } p \mid X)) ==> (\text{AP}(\text{LAMBDA } X(p \mid X))X) = (p \mid X)$
- A3. $(\text{AP}(\text{LAMBDA } X(p \mid X))X) = (\text{IF}(\text{BADLAMBDA } p \mid X)\text{NIL}(p \mid X))$
- T4. $(\text{AND}(\text{NOT}(\text{BADLAMBDA } p \mid X))$

(NOT(BADLAMBDA S))
 (NOT(LAMBDA S))) ==> (AP S X) = (EQUAL S X)

A5. (AND(NOT(BADLAMBDA S))
 (NOT(LAMBDA S))) ==> (AP S X) = (IF(BADLAMBDA X)NIL(EQUAL S X))

One axiom for determining the equality of functions is assumed:

A6. (EQUAL (LAMBDA v(p v))(LAMBDA v(q v))) =
 (ALL X(EQUAL(AP(LAMBDA v(p v))X) (AP(LAMBDA v(q v))X)))

One theorem (similar to T1 above) is assumed for use by the system:

T7. (AP F X) ==>(NOT(BADLAMBDA X))

Set theoretic abstraction: SET and ELEmenthood are defined as follows:

(SET v(p v)) = (LAMBDA v(p v))
 (ELE x s) = (APPLY s x)

Thus every function is a set and every set is a function. Functional equality is of course EQUAL, whereas set equality is:

(SETEQUAL x y) = (ALL V(IFF(ELE V x)(ELE V y)))

The set theory axioms are neutral with respect to a classical or Lesniewskian set theory. For example, at least the following axiom should be added for Lesniewskian set theory: (EQUAL(SET v(EQUAL v X))X).

MODAL LOGIC

The SYMMETRIC LOGIC also includes two modal operators: (QCL p) meaning: p is QCL-logically true (POS p) meaning: p is QCL-logically possible.

The QCL axioms are:

1. if x is NIL-or-CONTINGENT then (QCL x) = NIL
2. if x is not CONTINGENT then (QCL x) = x (i.e., x=T,NIL,or non-bool)

The POS axioms are:

1. if x=NIL then (POS x) = NIL
2. if x is not NIL then (POS x) = T

Other facilities for Modal Logic are described in (Brown17,18,21).

AXIOMATIZING THE THEORY

The QCL language may be augmented with new data structures, new function definitions, and other axioms. The function bodies of definitions may include quantifiers.

DATA STRUCTURES: THE SHELL PRINCIPLE

The shell principle allows QCL to create functions for dealing with new data structures after the user specifies a minimum of information. The Shell Principal produces axioms about data structures which obey the Fundamental Deduction Principle discussed in section 2. Although many of the axioms are similar to those described in (Boyer1), some of the axioms are, in order to preserve the property of replacing expressions by equal expressions, necessarily quite different. Shells are added via:

(SHELLCREATE const btm recog ac-list type-list default-list)

with

const	The name of the new function which constructs objects of the new type.
recog	The name of the new function which recognizes objects of the new type.
btm	Bottom object. Use T if there is no bottom object.
ac-list	List of the functions which are accessors of the data structure.
type-list	List of the type restrictions on the shell. Each restriction is an arbitrary formula in QCL consisting of symbols defined at that time.
default-list	List of the default values for the shell.

Shells may be tested for validity via:

(SHELLTEST const recog arity ac-list type-list default-list)

The function SHELL, combines the functions of SHELLTEST and SHELLCREATE. (To save computation time, it is recommended that SHELLCREATE be used instead, with the user calling SHELLTEST only when the shell is first introduced.)

Shells for the basic INTERLISP data structures of lists, positive integers, negative integers, decimal numbers, strings, and atoms are initially automatically created when QCL is entered by the commands:

```
(SHELLCREATE 'CONS 'T 'LISTP '(CAR CDR) '(T T) '(NIL NIL)
(SHELLCREATE 'PADDI '(ZERO) 'PNUMBERP '(PSUBI)
              '((PNUMBERP X1)) '((ZERO)))
```

```

(SHELLCREATE 'NMINUS 'T 'NNUMBERP '(POSPART)
  '(((IF(EQUAL X1 0) NIL(PNUMBERP X1)))) '(0))
(SHELLCREATE 'TENDIV 'T 'DECIMALP '(SIG MANTISSA)
  '(((AND(NOT(EQUAL X1 0))(OR(PNUMBERP X1)(NNUMBERP X1))))
  '(AND(PNUMBER X2)(NOT(EQUAL X2 0)))))
  '(1 1))
(SHELLCREATE 'CONCATPNUM '(NULLSTRING) 'STRINGP
  '(FIRST REST) '((PNUMBERP X1)(STRINGP X2))
  '(0 (NULLSTRING)))
(SHELLCREATE 'MAKEATOM 'T 'LITATOMP '(MAKESTRING)
  '(((AND(STRINGP X1)(NOT(MEMB X1 ("NIL","T"," ")))))) '("0"))

```

The shell axioms are qcl-true.

THEORY OF QUOTATION

Expressions concerning these six initial shells can be abbreviated by a theory of quotation based on normal INTERLISP conventions. The following are examples of the special shorthand equivalents for lists, strings and numbers which can save the user time and effort.

```

(QUOTE (THIS IS A LIST)) = (CONS (QUOTE THIS)
                                (CONS (QUOTE IS)
                                    (CONS (QUOTE A)
                                        (CONS (QUOTE LIST) NIL))))
4=(ADD1 (ADD1 (ADD1 (ADD1 (ZERO)))))
"ABC" = (CONCATPNUM (QUOTE 65)
                  (CONCATPNUM (QUOTE 66)
                              (CONCATPNUM (QUOTE 67)
                                            (QUOTE " "))))
(QUOTE ABC) = (MAKEATOM "ABC")

```

Note that numbers, strings, NIL (the normal end of a list), and T are automatically quoted. Sometimes C-LISP conventions will also work.

DEFINITIONS

Definitions are of the form:

(EQUAL (function-name-being-defined arg1...argN)definition-body)

The body of a definition may involve both recursive calls to themselves and quantifiers such as ALL, EX, LAMBDA. The symbol being defined may itself be a quantifier. Schemators may also appear in both the argument list and body of the function being defined. For example a recursive definition for the summation function, SIGMA, is:

```

(EQUAL (SIGMA K(SCHI K)M N)
  (IF (PNUMBERP M)
    (IF (PNUMBERP N)
      (IF (EQUAL M N)
        (SCHI N)
        (IF (LESSP M N)
          (PLUS (SIGMA K(SCHI K)M(PSUMI N))(SCHI N))
          0))
      0)0) )

```

Recursive definitions for CLISP iteration operators can also be defined using schemators. Such definitions for COLLECT and JOIN are given below:

```

(EQUAL (COLLECT K(SCHI K)M N)
  (IF (PNUMBERP M)
    (IF (PNUMBERP N)
      (IF (EQUAL M N)
        (LISTI(SCHI N))
        (IF (PLESSP M N)
          (APPEND (COLLECT K(SCHI K)M(PSUBI N))(LISTI (SCHI N)))
          NIL))
      NIL)NIL) )
(EQUAL (JOIN K(SCHI K)M N)
  (IF (PNUMBERP M)
    (IF (PNUMBERP N)
      (IF (EQUAL M N)
        (SCHI N)
        (IF (PLESSP M N)
          (APPEND (JOIN K(SCHI K)M(PSUBI N))(SCHI N))
          NIL))
      NIL)NIL) )

```

Note that SCHI is a schemator.

The commands which create definitions are:

(DEF definition) Adds a definition to those already in existence after ensuring that it is either an explicit or recursive definition. Induction templates are created and declared if the definition is recursive, and the type of the function value of the defined symbol is declared.

(DEFL list-of-definitions) Adds a list of definitions to those already existing using DEF.

(DEFLQ definition1...definitionN) FSUBR (FSUBR version of DEFL)

SYMEVALDEF Switch, when set to T causes the DEFinition commands to SYMbolically EVALuate the bodies of the definitions before the definitions are actually asserted. The default is T.

The following declaration commands can be used to assert definitions that the system is currently unable to define automatically:

(DCL definition) Declares a definition, and assumes that the definition is recursive on its first argument position. In fact it assumes that the COUNT measure is decreasing on the measured unit set consisting of the formal variable in that position. It does not first prove this fact and thus does not guard against infinite recursion. This command is useful for asserting definitions which are recursive but which are too difficult for this theorem prover to prove. Psuedo functions (i.e., functions which are executed for their side effects) should be declared so as to force their type calculation to occur at run time instead of at definition time.

(DCLL list.of.definitions) Declares a list of definitions using DCL.

(DCLLQ definition1...definitionN) FSUBR (FSUBR version of DCLL).

SYMEVALDCL Switch, when set to T causes the DeCLaration commands to SYMbolically EVALuate the bodies of the definitions before the definitions are actually asserted. The default is T.

Definition schemes are INTERLISP functions which when given an expression beginning with the symbol being defined returns another expression with that symbol replaced by its definition. These schemes are useful for defining new quantifiers including recursive quantifiers such as SIGMA, and infinite numbers of definitions. Definition schemes are implemented by use of the following command:

(DCLSCHEME symbol.being.defined interlisp.function.name)

The system does not automatically produce induction templates and type information for declared symbols. Such information can be given to the system by the following commands.

(DCLTEMPLATE symbol '(measure.expression machine axioms-justifying the template)). For example the template for SIGMA is: '(((COUNT M) (((LESSP M N) ((M (PSUB1 N)))) PSUB1.LESSP).

(ADDDTYPE symbol (list.of.types . list.of.argument.positions)). For example, the type of SIGMA is: '(PNUMBERP NNUMBERP DECIMALP OTHERN). NIL).

The following definitions are initially made when QCL is entered:

(DEFLQ

```
(EQUAL (LET V(SCH1 V)X) (SCH1 X))
(EQUAL (NOT P) (IF P NIL T))
(EQUAL (AND P Q) (IF P Q NIL))
(EQUAL (OR P Q) (IF P P Q))
(EQUAL (LOR P Q) (IF P T Q))
(EQUAL (IMPLIES P Q) (IF P Q T))
(EQUAL (IMPLY P Q) (IF P Q T))
(EQUAL (IFF P Q) (IF P(IF Q T NIL)(IF Q NIL T)))
(EQUAL (OBJECT X) (IF (BADLAMBDA P X) NIL X))
```

```

(EQUAL (LITATOM X) (LOR (LITATOMP X)(LOR (EQUAL X NIL)(EQUAL X T))))
(EQUAL (INUMBERP X) (OR (PNUMBERP X)(NNUMBERP X)))
(EQUAL (NUMBERP X) (OR (OTHERN X)(OR (DECIMALP X)(INUMBERP X))))
(EQUAL (LISP.DATA.STRUCTURE X)
      (LOR (LISTP X)(LOR (STRINGP X)(LOR (NUMBERP X)(LITATOM X)))))
)

```

The (LET V(SCHI V)X) function is used to assign a variable V to the result of SYMBolically EVALuating the value X only once and then substituting it into the (SCHI v) expression as many times as v occurs there. It is somewhat like (APPLY (LAMBDA V(SCHI V)) X) but without any comitment to X not being a BADLAMBDA. This construct is also analogous to the: "Let V=X and_return (SCHI v)" construct found in some programming languages.

Note that AND and OR are the normal INTERLISP AND and OR functions where AND returns the last true value or NIL, and OR returns the first true value or NIL. LOR is a more logical OR function returning T or the third argument. AND, OR, and LOR have the following properties:

These laws hold:

```

(AND (AND P Q) R) = (AND P (AND Q R))
(OR (OR P Q) R) = (OR P (OR Q R))
(LOR (LOR P Q) R) = (LOR P (LOR Q R))
(IF P X Y) = (OR (AND P X)(AND (NOT P) Y))

```

These laws do not hold:

```

(IF P X Y) = (AND (OR P Y)(OR (NOT P) X))
(IF P X Y) = (LOR (AND P X)(AND (NOT P) Y))
(IF P X Y) = (AND (LOR (NOT P) X)(LOR P Y))

```

REDUCTION AXIOMS

Reduction axioms are of one of the following forms

```

P replaces p by T
(EQUAL p q) replaces p by q
(IMPLIES c(EQUAL p q)) replaces p by q, whenever c holds
(IF c (EQUAL p q) T) replaces p by q, whenever c holds

```

Unlike definitions, reduction axioms should not involve infinite looping. In particular, the q expression should be simpler than p. For example, q should not contain an alphabetic variant of p.

(ADDAXIOM axiom name-of-axiom)

Adds axiom as a new rewrite rule to the system.

(ADDAXIOMS list-of-axioms) FSUBR

Calls ADDAXIOM on each axiom in the list. It uses the first symbol in the "p" sub-expression as the default name.

(ADDELIM elimination-axiom)

A special kind of reduction axiom schema called an elimination axiom may be added by this command. Elimination axioms have the form:

```
(EQUAL (h X Z1...Zn)
        (EX (I1...Im (AND (EQUAL (const Z1...Zn I1...Im) X)(c Z1...Zn I1...Im)))) )
```

For example:

```
(EQUAL (h X Z1...Zn)(EX I1...Im (EQUAL (const Z1...Zn I1...Im) X)) )
```

They have the effect of the rewrite schemas attached to the ALL and EX quantifiers:

```
(EQUAL (ALL X (s X))
        (AND(ALL Z1...(ALL Zn (ALL I1... (ALL In
                                         (IMPLIES (c Z1...Zn I1...Im)
                                         (s (const Z1...Zn I1...Im)) ) ))))
        (ALL Z1...(ALL Zn (ALL X (IMPLIES (NOT (h X Z1...Zn))(s X)))) ) )
(EQUAL (EX X (s X))
        (OR (EX Z1...(EX Zn (EX I1...(EX Im (AND (c Z1...Zn I1...Im)
                                         (s (const Z1...Zn I1...Im)) ) ))))
        (EX Z1...(EX Zn (EX X (AND (NOT (h Z1...Zn))(s X)))) ) )
```

The ELAXL list determines the use of such lemmas. It has the form:

```
( (elaxname { (selector { ...
               { (selector position) { ... } ) } ... )
```

(ADDRULE symbol lisp-function-name) The lisp-function-name is a lisp function which has the effect of one or more rewrite axioms associated with the formal symbol. This command is useful for axiomatizing quantifiers and other schematic symbols. It is also useful for asserting complex axiom schemes.

EQUATION SOLVING

SYMEVAL will try to solve equations at the appropriate time, and use those solutions at the appropriate time provided the user defines an INTERLISP function called RSOLVE which will solve equations for 0 or more solutions. The system will attempt to solve equations for variables whenever the quantifier binding that variable is being applied, or whenever that variable is free in the theorem being proven (i.e., whenever the variable is an unknown). The system will also attempt to solve for an appropriate variable of lowest scope in an equation occurring as the first argument of an IF function value provided that the function value also occurs in the second argument position of the IF statement.

INDUCTION AXIOMS

The induction scheme was adopted from (Boyer1) and has since been extended to handling recursive definitions containing quantifiers, bound variables and schemators. Induction axioms have the form:

```
(IMPLIES (test X1...Xn)
  (LESSP (measure (selector X1...Xn))(measure X1...Xn)) )
```

Many induction axioms are automatically created when the SHELL command is used to create a data structure. For example:

```
(IMPLIES (LISTP X)
  (LESSP (COUNT (CDR X))(COUNT X)) )
```

is created when the list data structure is asserted. Additional induction lemmas may be asserted with the following command:

```
(MKINAXIOM induction.axiom.name induction.axiom)
```

LINK TO INTERLISP INTERPRETER

The theorem prover automatically links to the INTERLISP interpreter whenever all the evaluated arguments of a function are explicit values or quoted objects provided that the function name appears on the FUNSYM list. This is done because INTERLISP is much faster than SYMEVAL. The SHELL, DEF, and DCL commands automatically place the function names they deal with onto the FUNSYM list. Thus the user should be careful to define any logical functions having the same name as some INTERLISP function, to exactly correspond in effect to that INTERLISP function. However, it is not required that logical definitions exist for a link to interlisp to be made. For example, if the name of every INTERLISP subr function were CONSed onto FUNSYM then QCL would be a superset of INTERLISP subrs. INTERLISP psuedo-functions (i.e., functions which are being executed for their side effects should be DeCLared (DCL) not DEfined, because DEF creates type info at definition time, which can cause the defined not to be executed at run time. For example, if the type of a DEfined print function were NIL it would never be executed at run time because the system would know that the function value equaled NIL.

USING THE SYSTEM

Anyone who uses this system for programming as opposed to more general deductive tasks should bear in mind that this is a general theorem proving system which has not been engineered especially for computation. Thus, the system will appear to be somewhat slow

in comparison to systems engineered specifically for computational tasks. This slowness is in no way a reflection on how well an interpreter for this language could be engineered for computational tasks.

INITIALIZING THE SYSTEM

getting started on the Research DEC20.

```
.LISP  
*LOAD (AUX:◀CS.BROWN▶ATP.COM)
```

restarting the system:

```
(*SYSINIT)
```

The system version may be obtained by typing

```
ATP.VERSION
```

This paper corresponds to version 3.

EXECUTING PROGRAMS AND PROVING THEOREMS

(PV theorem) FSUBR (but if theorem is a variable it is evaluated like a SUBR) attempts to prove a theorem using PV and collects statistics on the process. More useful than PROVE because theorem could be a variable containing the theorem.

(PROVE theorem) Attempts to prove a theorem using recursive SYMBolic EVALuation: SYMEVAL0, and Noetherian induction.

(SYMEVAL0 theorem) Attempts to prove a theorem using only recursive SYMBolic EVALuation.

DEBUGGING PROGRAMS

As SYMEVAL0 recursively evaluates expressions tracing information is produced whenever a definition, axiom, or rule is applied. This information consists of three parts: (1) An input expression to which the axiom is being applied, called I. (2) The midterm expression produced by the application of the axiom, called M. (3) The output expression obtained by recursively evaluating the M expression, called O. Thus, a trace will generally be in the form:

```
I:exp  
M:exp
```

```

    l2:exp
    M2:exp
    ...
    O2:exp
    l2:exp
    M2:exp
    ...
    O2:exp
    O1:exp

```

where the numbers immediately following l, M, or O are the level at which that application takes place. At a given level number i, O_i and M_i are always associated with the preceding l_i.

DEBUGGING PRINCIPLE The basic method of debugging is this: if some l_i expression is not logically equal to the corresponding O_i in the given theory and in the specific context, then either l_i does not equal M_i or M_i does not equal O_i. If l_i does not equal M_i then the definition, axiom, or rule used in that application is incorrect. If M_i does not equal O_i then one must recursively examine the i+1 level (i.e., l_{i+1}, M_{i+1}, O_{i+1}) to find the error.

TRACELIST List of currently defined symbols to be traced. Each time one of these symbols is used, tracing information will be printed in the trace file. (Note: if there is no trace file, the tracing information defaults to the screen.) When a trace of all function symbols is desired, TRACELIST can be set to FUNSYM.

OPENTR (trace-file) FSUBR Sets up a file for tracing information.

CLOSETR() Closes the current trace file.

TRACEM Switch, when set to T, causes the mid term (the term after an expression has been replaced by its definition, but before it has been evaluated) to be printed as well as the input and output expressions on the trace. The default is T.

TRACEN Switch, when set to T, causes the name of each axiom or rule to be printed when it is used. The default is T.

TRACEH Switch, when set to T, causes the evaluation of any hypothesis to an axiom, which the system is trying to use, to be traced. The default is NIL.

TRACESYS Switch, when set to T, causes the a-list and type set a-list to be printed along with each tracing line. The default is NIL.

EXAMPLE OF USING THE SYSTEM

Here is a simple example of using SYMEVAL-QCL as a programming language for natural language processing. We define functions for recognizing an adjective list and translating it into logic. For our purposes, an adjective list is either NIL, or an adjective followed by an adjective list.

©lisp

INTERLISP-10 27-NOV-79 ...

Hello, Handsome.

2_LOAD(ATP.COM)

compiled on 7-Jun-82 10:13:24

(ADDRULE redefined)

Transoring of<(CS.BROWN>NATP...17

done on 13-Feb-81 12:50:41

<CS.BROWN>ATP.COM.1

3_LOAD(ATP.DEFS)

<CS.BROWN>ATP.DEFS.1

After loading the theorem prover and the ATP.DEFS environment, we define SASSOCP for looking up words in the lexicon.

```
4_(DEFLQ
  (EQUAL (SASSOCP X L V)
    (EX U
      (IF (EQUAL U (SASSOC X L))
        (IF U (EQUAL V U) NIL)
        NIL))))
```

NIL

Next, we define the lexicon.

```
5_(DEFLQ
  (EQUAL (ADJW)
    (QUOTE ((RED RED)
      (BIG BIG)
      (SMALL SMALL))))
```

NIL

Now, we define ADJ for processing a single adjective.

```
6_(DEFLQ
  (EQUAL (ADJ X Y A Z)
    (EX U
      (IF (SASSOCP (CAR X)(ADJW) U)
        (IF (EQUAL Y (CDR X))
          (EQUAL Z (LIST2 (CAR (CDR U)) A))
          NIL)
        NIL))))
```

NIL

Finally, we define ADJL to process an adjective list.

```

7_(DCLLQ
  (EQUAL (ADJL X0 X2 A Z)
    (IF (EX X1
      (EX Z1
        (IF (ADJ X0 X1 A Z1)
          (EX Z2
            (IF (ADJL X1 X2 A Z2)
              (EQUAL Z (LIST3 (QUOTE AND) Z1 Z2))
              NIL))
          NIL)))
    T
    (IF (EQUAL Z T) (EQUAL X2 X0) NIL))))
NIL

```

Having defined ADJL, we can now test it.

```

8_(PV (ADJL (QUOTE ()) NIL (QUOTE A) Z))
  (EQUAL Z T)

9_(PV (ADJL (QUOTE (BIG RED)) NIL (QUOTE A) Z))
  (EQUAL Z (QUOTE (AND (BIG A) (AND (RED A) T))))

10_(PV (ADJL (QUOTE (BIG BOY)) (QUOTE (BOY)) (QUOTE A) Z))
  (EQUAL Z (QUOTE (AND (BIG A) T)))

```

RULE PACKAGES

These libraries of rules may be accessed from account AUX:<CS.BROWN>. For example: AUX:<CS.BROWN>ATP.DEFS accesses the rule package ATP.DEFS described below.

ATP.COM This is the basic theorem prover code. It automatically sets up an initial environment for QCL programming.

REAL ALGEBRA

ATP.REAL This file contains a theory for simplifying expressions of real algebra. The primitive symbols are:

```

(PLUS n m)
(MINUS n)
(TIMES n m)
(EXP n i)
(LESSP n m)

```


This file contains rules axiomatizing each of these primitive symbols. A description of these rules is given in (Brown24). These rules include rewrite rules dealing with algebraic simplification and basis argument rule for eliminating ALL before an EQUAL expression. It also includes the equation solving system: RSOLVE linked to via SYMEVALS equation solving link. The defined symbols of the theory are:

(ADD1 n)
 (SUB1 n)
 (DIFFERENCE n m)
 (QUOTIENT n m)
 (MINUSP n)
 (GREATERP n m)
 (LEQ n m)
 (GEQ n m)
 (MIN n m)
 (MAX n m)
 (ABS n)
 (SQRT n)
 (FAC n) factorial
 (CO x y) combinations
 (SIGMA :k(f k)m n) sigma

MISCELLANEOUS DEFINITIONS

ATP.DEFS Defines some basic recursive functions for QCL programming along the lines described in (Brown20,Kowalski).

1. Defines more functions for manipulating lists:
 CADDR CDDR CAAR CDAR CADDR CDDDR CADDDR LIST1 LIST2
 LIST3 NLISTP PLISTP APPEND SASSOC REVERSE MEMBER OCCUR
2. Defines functions for positive integers:
 PFIX PZEROP PPLUS PTIMES PDIFFERENCE PHALF PLESSP GCG
 INUMBERP

ATP.SORT Contains various sorting functions including a merge sort and two quicksorts.

SCHWIND'S THEORY OF LANGUAGE ANALYSIS

ATP.NATL A fragment of Schwind's theory of language analysis (Schwind,Brown,13,14,15). This fragment of Schwind's theory specifies how a small subset of English may be translated into the QCL Logic. The most important function in ATP.NATL is TEXT. TEXT takes a piece of text (one or more sentences) and returns its representation in logic. There are a number of lower level functions for translating nouns, verbs, prepositional phrases, relative clauses and so on. Consult ATP.NATL itself for these functions.

TEXT is invoked in the following manner.

(TEXT i-eng o-eng i-nlist i-vlist o-vlist o-logic)

with

i-eng The entire English input.

o-eng The remainder of the English input after the first sentence or piece of text has been processed.

i-nlist Input list of previously seen nouns for noun/pronoun substitution.

o-nlist Output list of previously seen nouns including any found during processing.

i-vlist Input list of object language variables.

o-vlist Output list of unused object language variables.

o-logic Output translation of the sentence or text into logic.

NATL A list of all the grammatical functions in NATL. Usually used to trace the parsing of a sentence, without tracing every FUNSYM.

We believe that other logically based natural language theories such as (Simmons1,2) would be naturally expressed in QCL and executed by SYMEVAL.

EXAMPLE DEFINITIONS

A few example definitions from Schwind's Natural Language Theory are given below. The first definition is a "backtracking" definition which involves a search through these 4 alternative non-exclusive cases. This definition states that a piece of text is a Noun Phrase if it is a Noun Phrase of type 0, 2, 3, or 4.

```
(EQUAL (NP X0 X1 N1 N2 V1 V2 A STAR Z FR)
  (IF (NP0 X0 X1 N1 N2 V1 V2 A STAR Z FR) T
    (IF (NP2 X0 X1 N1 N2 V1 V2 A STAR Z FR) T
      (IF (NP3 X0 X1 N1 N2 V1 V2 A STAR Z FR) T
        (NP4 X0 X1 N1 N2 V1 V2 A STAR Z FR) ))) )
```

The next definition is a recursive definition that states that a Noun Phrase List is either a Noun Phrase List or null.

```
(EQUAL (NPL X0 X2 N1 N3 V1 V3 AL STAR Z FRL)
  (IF (IF (LISTP X0)
    (EX X1 (EX N2(EX V2(EX A(EX Z2(EX FR
```

```

      (IF (NP X0 X1 N1 N2 V1 V2 A Z2 ZFR)
      (EX AL2 (EX FRL2 (IF (NPL X1 X2 N2 N3 V2 V3 AL2 STAR Z2 FRL2)
      (IF (EQUAL AL (CONS A AL2))
      (EQUAL FRL (CONS FR FRL2))
      NIL)
      NIL) ))
      NIL) )))))
    NIL)
  T
  (IF (EQUAL N3 N1)
  (IF (EQUAL Z STAR)
  (IF (EQUAL AL NIL)
  (IF (EQUAL X0 X2)
  (IF (EQUAL V1 V3)
  (EQUAL FRL NIL)
  NIL) NIL) NIL) NIL) NIL) ))

```

This definition states that the only verb groups according to this grammar are verbs.

```

(EQUAL (VG X Y V Z FR)
(VERB X Y V Z FR))

```

This last definition states that a piece of text is a DeCLarative SENTence TRANsitive if it is a Noun Phrase followed by a Verb Group followed by a Noun Phrase List.

```

(EQUAL (DCLSENTTRAN X0 X3 N0 N2 V0 V2 Z)
(EX X1(EX X2(EX N1(EX V1(EX A(EX AL(EX FR5(EX STAR(EX STAR2
(IF (NP X0 X1 N0 N1 V0 V1 A STAR2 Z FR5)
(EX FRV
(IF (VG X1 X2 (CONS A AL) STAR FRV)
(EX FRL2
(NPL X2 X3 N1 N2 V1 V2 AL STAR STAR2 FRL2))
NIL)) NIL))))))))) )

```

EXAMPLE EXECUTION

The English sentence "SOME BOY THROWS THE BIG RED BALL IN TEXAS" is proven to be a DeCLarative SENTence TRANsitive. In the course of this proof SYMEVAL deduces that there is exactly one possible translation of this sentence into QCL augmented with a special THE function. This translation is the expression equal to Z in the result of the evaluation given below.

It is worthwhile noting that this proof involves at least 201 existentially quantified variables, and that SYMEVAL systematically eliminates each one of these quantifiers. The final result contains no quantifiers or defined symbols, but is logically equivalent in Schwind's theory to the original input expression.

The expression to be recursively simplified is:

(DCLSENTTRAN (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))
NIL NIL N2 (QUOTE (X2 X3))
V3 Z)

11:(DCLSENTTRAN (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))
NIL NIL N2 (QUOTE (X2 X3))
V3 Z)

by use of: DCLSENTTRAN

M1:(EX

*1

(EX

*2

(EX

*3

(EX

*4

(EX

*5

(EX *6

(EX *7

(EX *8

(EX *9

(IF (NP (QUOTE (SOME BOY THROWS THE BIG RED
BALL IN TEXAS))

*1 NIL *3 (QUOTE (X2 X3))

*4 *5 *9 Z *7)

(EX *10

(IF (VG *1 *2 (CONS *5 *6)

*8 *10)

(EX *11

(NPL *2 NIL *3 N2 *4 V3 *6 *8 *9

*11))

NIL))

NIL)))))))))

12:(NP (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))

*1 NIL *3 (QUOTE (X2 X3))

*4 *5 *9 Z *7)

by use of: NP

M2:(IF (NPO (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))

*1 NIL *3 (QUOTE (X2 X3))

*4 *5 *9 Z *7)

T

(IF (NP2 (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))

*1 NIL *3 (QUOTE (X2 X3))

*4 *5 *9 Z *7)

T

(IF (NP3 (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))

*1 NIL *3 (QUOTE (X2 X3))

*4 *5 *9 Z *7)

T

(NP4 (QUOTE (SOME BOY THROWS THE BIG RED BALL IN TEXAS))

*1 NIL *3 (QUOTE (X2 X3))

```

                                *4 *5 *9 Z *7)))
02:(IF
  (EQUAL *7 (QUOTE (M THIRD)))
  (IF
    (EQUAL *1 (QUOTE (THROWS THE BIG RED BALL IN TEXAS)))
    (IF
      (EQUAL *4 (QUOTE (X3)))
      (IF
        *3 NIL
        (IF (EQUAL *5 (QUOTE X2))
          (EQUAL Z
            (CONS (QUOTE EX)
              (CONS (QUOTE X2)
                (CONS (CONS (QUOTE AND)
                  (CONS (QUOTE (BOY X2))
                    (CONS *9 NIL))))
                NIL))))))
        NIL))
      NIL)
    NIL)
  NIL)
12:(VG (QUOTE (THROWS THE BIG RED BALL IN TEXAS))
  *2
  (CONS (QUOTE X2)
    *6)
    *8 *10)
  by use of: VG
M2:(VERB (QUOTE (THROWS THE BIG RED BALL IN TEXAS))
  *2
  (CONS (QUOTE X2)
    *6)
    *8 *10)
02:(IF (EQUAL *2 (QUOTE (THE BIG RED BALL IN TEXAS)))
  (IF (EQUAL *8 (CONS (QUOTE THROW1)
    (CONS (QUOTE X2)
      *6)))
    (EQUAL *10 (QUOTE (AGENT DO DEST)))
    NIL)
  12:(NPL (QUOTE (THE BIG RED BALL IN TEXAS))
    NIL NIL N2 (QUOTE (X3))
    V3 *6 (CONS (QUOTE THROW1)
      (CONS (QUOTE X2)
        *6))
      *9 *11)
    by use of: NPL
M2:(IF
  (IF
    (LISTP (QUOTE (THE BIG RED BALL IN TEXAS)))
    (EX
      *63
      (EX

```

```

*64
(EX
  *65
  (EX
    *66
    (EX
      *67
      (EX
        *68
        (IF (NP (QUOTE (THE BIG RED BALL IN TEXAS))
          *63 NIL *64 (QUOTE (X3))
          *65 *66 *67 *9 *68)
        (EX *69
          (EX *70
            (IF (NPL *63 NIL *64 N2 *65 V3 *69
              (CONS (QUOTE THROW1)
                (CONS (QUOTE X2)
                  *6))
              *67 *70)
            (IF (EQUAL *6 (CONS *66 *69))
              (EQUAL *11 (CONS *68 *70))
              NIL)
            NIL)))
          NIL))))))
        NIL)
      T
      (IF (EQUAL N2 NIL)
        (IF (EQUAL *9 (CONS (QUOTE THROW1)
          (CONS (QUOTE X2)
            *6)))
          (IF (EQUAL *6 NIL)
            (IF (EQUAL (QUOTE (THE BIG RED BALL IN TEXAS))
              NIL)
              (IF (EQUAL (QUOTE (X3))
                V3)
                (EQUAL *11 NIL)
                NIL)
              NIL)
            NIL)
          NIL)
        NIL)
      NIL)
    NIL)
  NIL)
02:(IF
  (EQUAL N2 (QUOTE (((N THIRD ) . TEXAS))))
  (IF
    (EQUAL *9 (CONS (QUOTE THROW1)
      (CONS (QUOTE X2)
        *6)))
    (IF V3 NIL
      (IF (EQUAL *6
        (QUOTE ((THE X3 ((SING-PLURAL)
          (AND (AND (BIG X3)

```

```

                                (AND (RED X3)
                                T))
                                (AND (BALL X3)
                                (AND (IN X3 TEXAS)
                                T))))))
(EQUAL *11 (QUOTE ((N THIRD))))
NIL))
NIL)
NIL)
01:(IF
(EQUAL
Z
(QUOTE (EX X2
(AND (BOY X2)
(THROW1 X2
(THE X3 ((SING-PLURAL)
(AND (AND (BIG X3)
(AND (RED X3)
T))
(AND (BALL X3)
(AND (IN X3 TEXAS)
T))))))))))
(IF (EQUAL N2 (QUOTE (((N THIRD) . TEXAS))))
(IF V3 NIL T)
NIL)
NIL)

```

The result of recursive simplification is:

```

(IF
(EQUAL
Z
(QUOTE (EX X2 (AND (BOY X2)
(THROW1 X2
(THE X3 ((SING-PLURAL)
(AND (AND (BIG X3)
(AND (RED X3)
T))
(AND (BALL X3)
(AND (IN X3 TEXAS)
T))))))))))
(IF (EQUAL N2 (QUOTE (((N THIRD) . TEXAS))))
(IF V3 NIL T)
NIL)
NIL)

```

end of deduction

EX1=110/EX2=0/EX3=71/EX4=0/EX5=6/EX6=0/EX7=0/EX8=0/EX9=0/EX10=0/
ALL1=0/ALL2=0/ALL3=0/ALL4=0/ALL5=0/ALL6=0/ALL7=0/ALL8=0/ALL9=0/ALL10=0/

SET THEORY

Axioms for LAMBDA abstraction are part of the SYMEVAL-QCL system and are described in section 3.5. Set Theory is developed in terms of LAMBDA abstraction by first defining set theoretic abstraction and elementhood in terms of LAMBDA abstraction and APPLYcation, and then by asserting which sets are good sets in the sense that they may be members of other sets.

ATP.SETDEF

This file contains definitions and theorems for the set theory described in Quine's book: SET THEORY AND ITS LOGIC.

ATP.SET

Extra code for implementing set theory. Not currently used.

EXAMPLE

SYMEVAL can prove that the Weiner-Kurtowski set theoretic definition of an ordered pair is in fact an ordered pair. This proof is obtained without the use of any lemmas whatsoever, and in fact in the course of the proof SYMEVAL proves a number of interesting lemmas about the equality of unordered pairs and unit sets. The only other automatic proof of this theorem that we are aware of in the literature is the sequent calculus based proof in (Brown6) which assumed several lemmas about unordered pairs and unitsets. (That sequent calculus theorem prover could prove the lemmas that were assumed if they were explicitly given to it.)

In order to state the ordered pair theorem, quantifiers whose bound variables range over anything but bad sets are declared

```
(SETQ QUANTSYM(APPEND '(QALL QEX SET) QUANTSYM))
(VASSUME 'SET (REMOVE 'BADLAMBDAP UNIVERSE))
(VASSUME 'QALL (REMOVE 'BADLAMBDAP UNIVERSE))
(VASSUME 'QEX (REMOVE 'BADLAMBDAP UNIVERSE))
```

and then the following definitions are made:

```
(DCLLQ(EQUAL (ELE X Y) (AP X)))
      (EQUAL (SET X(SCHI X)) (LAMBDA X(SCHI X)))
      (EQUAL (QALL X(SCHI X)) (ALL X(IF (BADLAMBDAP X) T (SCHI X))))
```


(EQUAL (QEX X(SCH2 X)) (EX X(AND (NOT(BADLAMBDA X))(SCH2 X))))
 (EQUAL (EQUALSETS A B) (QALL X(IFF(ELE X A)(ELE X B)))))
 (EQUAL (UNITSET A) (SET X(EQUAL X A)))
 (EQUAL (PAIRSET A B) (SET X (LOR(EQUAL X A)(EQUAL X B))))
 (EQUAL (ORDPAIRSET A B) (PAIRSET(UNITSET A)(PAIRSET A B)))

Two axioms of set theory are assumed, namely that unit sets and unordered pairsets are not BADLAMBDA's:

AX5: (EQUAL(BADLAMBDA(LAMBDA X(EQUAL X A))) NIL)
 AX6: (EQUAL(BADLAMBDA(LAMBDA X(IF(EQUAL X A)T(EQUAL X B)))) NIL)

The ordered pair theorem: that two ordered pairs are equalsets of their components are equal is now proven.

The expression to be recursively simplified is:

(QALL X (QALL Y (QALL U (QALL V (IFF (EQUALSETS (ORDPAIRSET X Y)
 (ORDPAIRSET U V))
 (AND (EQUAL X U)
 (EQUAL Y V)))))))

I1:(ORDPAIRSET X Y)

by use of: ORDPAIRSET

M1:(PAIRSET (UNITSET X)
 (PAIRSET X Y))

I2:(UNITSET X)

by use of: UNITSET

M2:(SET *1 (EQUAL *1 X))

O2:(LAMBDA *1 (EQUAL *1 X))

I2:(PAIRSET X Y)

by use of: PAIRSET

M2:(SET *2 (LOR (EQUAL *2 X)
 (EQUAL *2 Y)))

O2:(LAMBDA *2 (IF (EQUAL *2 X)

T

(EQUAL *2 Y)))

I2:(PAIRSET (LAMBDA *1 (EQUAL *1 X))
 (LAMBDA *2 (IF (EQUAL *2 X)

T

(EQUAL *2 Y))))

by use of: PAIRSET

M2:(SET *3 (LOR (EQUAL *3 (LAMBDA *1 (EQUAL *1 X))
 (EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)

T

(EQUAL *2 Y))))))

O2:(LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1 (EQUAL *1 X))

T

(EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)

T

(EQUAL *2 Y))))))

01:(LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1 (EQUAL *1 X)))
^T
(EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)
^T
(EQUAL *2 Y))))))

11:(ORDPAIRSET U V)

by use of: ORDPAIRSET

M1:(PAIRSET (UNITSET U)
(PAIRSET U V))

I2:(UNITSET U)

by use of: UNITSET

M2:(SET *4 (EQUAL *4 U))

O2:(LAMBDA *4 (EQUAL *4 U))

I2:(PAIRSET U V)

by use of: PAIRSET

M2:(SET *5 (LOR (EQUAL *5 U)
(EQUAL *5 V)))

O2:(LAMBDA *5 (IF (EQUAL *5 U)
^T
(EQUAL *5 V)))

I2:(PAIRSET (LAMBDA *4 (EQUAL *4 U))
(LAMBDA *5 (IF (EQUAL *5 U)
^T
(EQUAL *5 V))))

by use of: PAIRSET

M2:(SET *6 (LOR (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
(EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
^T
(EQUAL *5 V))))))

O2:(LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
^T
(EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
^T
(EQUAL *5 V))))))

01:(LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4 (EQUAL *4 U)))
^T
(EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
^T
(EQUAL *5 V))))))

11:(EQUALSETS (LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1 (EQUAL *1 X)))
^T
(EQUAL *3 (LAMBDA *2 (IF (EQUAL *2 X)
^T
(EQUAL *2 Y))))))
(LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4 (EQUAL *4 U))
^T
(EQUAL *6 (LAMBDA *5 (IF (EQUAL *5 U)
^T
(EQUAL *5 V))))))

by use of: EQUALSETS

M1:(QALL *7 (IFF (ELE *7 (LAMBDA *3 (IF (EQUAL *3 (LAMBDA *1
 (EQUAL *1 X)))
 T
 (EQUAL *3 (LAMBDA
 *2
 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y))))))
 (ELE *7 (LAMBDA *6 (IF (EQUAL *6 (LAMBDA *4
 (EQUAL *4 U)))
 T
 (EQUAL *6 (LAMBDA
 *5
 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))))))))

I2:(IFF (IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
 T
 (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y))))))
 (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
 T
 (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))))

by use of: IFF

M2:(IF (IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
 T
 (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y))))))
 (IF (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
 T
 (EQUAL *7 LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))))
 T NIL)
 (IF (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
 T
 (EQUAL *7 LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))))

NIL T))

I3:(EQUAL (LAMBDA *1 (EQUAL *1 X))
 (LAMBDA *4 (EQUAL *4 U)))

by use of: (LISPLINK SYMEQUAL)

M3:(ALL *8 (EQUAL (AP (LAMBDA *1 (EQUAL *1 X))
 *8)
 (AP (LAMBDA *4 (EQUAL *4 U))
 *8)))

O3:(EQUAL U X)
 I3:(EQUAL (LAMBDA *1 (EQUAL *1 X))
 (LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))
 by use of: (LISPLINK SYMEQUAL)
 M3:(ALL *9 (EQUAL (AP (LAMBDA *1 (EQUAL *1 X))
 *9)
 (AP (LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))
 *9)))

O3:NIL
 I3:(EQUAL (LAMBDA *2 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y)))
 (LAMBDA *4 (EQUAL *4 U)))
 by use of: (LISPLINK SYMEQUAL)
 M3:(ALL *10 (EQUAL (AP (LAMBDA *2 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y)))
 *10)
 (AP (LAMBDA *4 (EQUAL *4 U))
 *10)))

O3:(IF (EQUAL U X)
 (EQUAL Y X)
 NIL)
 I3:(EQUAL (LAMBDA *2 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y)))
 LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))
 by use of: (LISPLINK SYMEQUAL)
 M3:(ALL *11 (EQUAL (AP (LAMBDA *2 (IF (EQUAL *2 X)
 T
 (EQUAL *2 Y)))
 *11)
 (AP (LAMBDA *5 (IF (EQUAL *5 U)
 T
 (EQUAL *5 V))))
 *11)))

O3:(IF (EQUAL X U)
 (IF (EQUAL Y U)
 (EQUAL V U)
 (EQUAL Y V))
 (IF (EQUAL X V)
 (IF (EQUAL Y V)
 NIL
 (EQUAL Y U))
 NIL))

```

O2:(IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
      (EQUAL U X)
      (IF (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
                                     T
                                     (EQUAL *2 Y))))
          (IF (EQUAL U X)
              (IF (EQUAL Y X)
                  T
                  (EQUAL Y V))
              (IF (EQUAL X V)
                  (IF (EQUAL Y V)
                      NIL
                      (EQUAL Y U))
                  NIL))
          (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
              NIL
              (IF (EQUAL *7 (LAMBDA *5 (IF (EQUAL *5 U)
                                             T
                                             (EQUAL *5 V))))
                  NIL T))))))

I2:(QALL *7 (IF (EQUAL *7 (LAMBDA *1 (EQUAL *1 X)))
                (EQUAL U X)
                (IF (EQUAL *7 (LAMBDA *2 (IF (EQUAL *2 X)
                                             T
                                             (EQUAL *2 Y))))
                    (IF (EQUAL U X)
                        (IF (EQUAL Y X)
                            T
                            (EQUAL Y V))
                        (IF (EQUAL X V)
                            (IF (EQUAL Y V)
                                NIL
                                (EQUAL Y U))
                            NIL))
                    (IF (EQUAL *7 (LAMBDA *4 (EQUAL *4 U)))
                        NIL
                        (IF (EQUAL *7 LAMBDA *5 (IF (EQUAL *5 U)
                                                       T
                                                       (EQUAL *5 V))))
                            NIL T))))))

by use of: QALL
M2:(ALL *12
    (IF (BADLAMBDA *12)
        T
        (IF (EQUAL *12 (LAMBDA *1 (EQUAL *1 X)))
            (EQUAL U X)
            (IF (EQUAL *12 (LAMBDA *2 (IF (EQUAL *2 X)
                                           T
                                           (EQUAL *2 Y))))
                (IF (EQUAL U X)
                    (IF (EQUAL Y X)

```

```

      T
      (EQUAL Y V))
    (IF (EQUAL X V)
      (IF (EQUAL Y V)
        NIL
        (EQUAL Y U))
      NIL))
  (IF (EQUAL *12 (LAMBDA *4 (EQUAL *4 U)))
    NIL
    (IF (EQUAL *12 (LAMBDA *5 (IF (EQUAL *5 U)
      T
      (EQUAL *5 V))))
      NIL T))))))
I3:(BADLAMBDA (LAMBDA *1 (EQUAL *1 X)))
  by use of: AX5
M3:NIL
O3:NIL
I3:(BADLAMBDA (LAMBDA *2 (IF (EQUAL *2 X)
  T
  (EQUAL *2 Y))))
  by use of: AX6
M3:NIL
O3:NIL
I3:(BADLAMBDA (LAMBDA *4 (EQUAL *4 X)))
  by use of: AX5
M3:NIL
O3:NIL
I3:(BADLAMBDA (LAMBDA *5 (IF (EQUAL *5 X)
  T
  (EQUAL *5 V))))
  by use of: AX6
M3:NIL
O3:NIL
I3:(BADLAMBDA (LAMBDA *4 (EQUAL *4 X)))
  by use of: AX5
M3:NIL
O3:NIL
I3:(BADLAMBDA (LAMBDA *5 (IF (EQUAL *5 X)
  T
  (EQUAL *5 V))))
  by use of: AX6
M3:NIL
O3:NIL
O2:(IF (EQUAL U X)
  (IF (EQUAL Y X)
    (EQUAL V X)
    (EQUAL Y V))
  NIL)
O1:(IF (EQUAL U X)
  (IF (EQUAL Y X)
    (EQUAL V X)

```

```

(EQUAL Y V))
NIL)
II:(IFF (IF (EQUAL U X)
             (IF (EQUAL Y X)
                 (EQUAL V X)
                 (EQUAL Y V))
             NIL)
         (IF (EQUAL X U)
             (EQUAL Y V)
             NIL))
by use of: IFF
MI:(IF (IF (EQUAL U X)
            (IF (EQUAL Y X)
                (EQUAL V X)
                (EQUAL Y V))
            NIL)
        (IF (IF (EQUAL X U)
                (EQUAL Y V)
                NIL)
            T NIL)
        (IF (IF (EQUAL X U)
                (EQUAL Y V)
                NIL)
            NIL T))
OI:T
II:(QALL V T)
by use of: QALL
MI:(ALL *21 (IF (BADLAMBDA *21)
                T T))
OI:T
II:(QALL U T)
by use of: QALL
MI:(ALL *22 (IF (BADLAMBDA *22)
                T T))
OI:T
II:(QALL Y T)
by use of: QALL
MI:(ALL #23 (IF (BADLAMBDA *23)
                T T))
OI:T
II:(QALL X T)
by use of: QALL
MI:(ALL *24 (IF (BADLAMBDA *24)
                T T))
OI:T

```

The result of recursive simplification is:
T

which is true. QED.

EX1=0/EX2=0/EX3=0/EX4=0/EX5=0/EX6=0/EX7=0/EX8=0/EX9=0/EX10=0/
ALL1=0/ALL2=0/ALL3=22/ALL4=0/ALL5=0/ALL6=10/ALL7=0/ALL8=0/ALL9=0/ALL10=0/

ONTOLOGY

Lesniewski's Ontology (Luschei, Henry) is a set theory which grew out of the traditions of Medieval logic. Its "sets" closely correspond to noun phrases, including names, fictitious names (e.g., Pegasus) and more general nouns. Its "elementhood" predicate corresponds to the intransitive verb IS in English, and more closely to the Latin EST.

ATP.LES

This file contains some definitions for Lesniewski's theory of Ontology.

EXAMPLE

SYMEVAL can prove that '(Z X Y) is a permutation of '(X Y Z). In order to do this, we first define recursive ontological definitions of the notion of a permutation:

```
(DCLLQ
(EQUAL (PERMSET L) (IF (EQUAL L (NILSET))
                        NIL
                        (INSERTSET (CAR L) (PERMSET (CDR L)))))
(EQUAL (NILSET) (LAMBDA X (EQUAL X NIL)))
(EQUAL (CONSET A B)
  (LAMBDA X (EX Y (EX Z (AND (IS Y A) (AND (IS Z B) (EQUAL X (CONS Y Z)))))))
(EQUAL (INSERTSET X L)
  (IF (EQUAL L (NILSET))
      (CONS X NIL)
      (NOMINAL.OR (CONSET X L)
                   (LAMBDA Y (EX Z (AND (IS Z L)
                                         (IS Y (CONSET (CAR Z)
                                                         (INSERTSET X (CDR Z))))))))))
(EQUAL (NOMINAL.OR A B)
  (LAMBDA X (LOR (IS X A) (IS X B)))))
(SETQ TRACELIST '(NOMINAL.OR CONSET PERMSET INSERTSET))
```

A proof of the Ontological theorem:

```
(IS (QUOTE(Z Y X)) (PERMSET(QUOTE(X Y Z))))
```

is given below. The proof was edited by deleting most traces less than level 2.

The expression to be recursively simplified is:


```

(IS (QUOTE (Z Y X))
  (PERMSET (QUOTE (X Y Z))))
I1:(PERMSET (QUOTE (X Y Z)))
  by use of: PERMSET
M1:(IF (EQUAL (QUOTE (X Y Z))
              (NILSET))
  NIL
  (INSERTSET (CAR (QUOTE (X Y Z)))
              (PERMSET (CDR (QUOTE (X Y Z))))))
I2:(PERMSET (QUOTE (Y Z)))
  by use of: PERMSET
M2:(IF (EQUAL (QUOTE (Y Z))
              (NILSET))
  (NIL
    (INSERTSET (CAR (QUOTE (Y Z)))
                (PERMSET (CDR (QUOTE (Y Z))))))
  I3:(PERMSET (QUOTE (Z)))
    by use of: PERMSET
    M3:(IF (EQUAL (QUOTE (Z))
                  (NILSET))
      NIL
      (INSERTSET (CAR (QUOTE (Z)))
                  (PERMSET (CDR (QUOTE (Z))))))
    I4:(PERMSET NIL)
      by use of: PERMSET
      M4:(IF (EQUAL NIL (NILSET))
        NIL
        (INSERTSET (CAR NIL)
                    (PERMSET (CDR NIL))))
    04:NIL
    03:(QUOTE (Z))
    I3:(INSERTSET (QUOTE Y)
                  (QUOTE (Z)))
      by use of: INSERTSET
      M3:(IF (EQUAL (QUOTE (Z))
                    (NILSET))
        (CONS (QUOTE Y)
              NIL)
        (NOMINAL.OR (CONSET (QUOTE Y)
                             (QUOTE (Z)))
                     (LAMBDA
                      *8
                      (EX *9 (AND (IS *9 (QUOTE (Z)))
                                   (IS *8 (CONSET (CAR *9)
                                                  (INSERTSET (QUOTE Y)
                                                                (CDR *9))))))))
        )
    03:(LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
                              *20)

```

T

```

(EQUAL (QUOTE (Z Y))
*20)))
02:(LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
*20)
T
(EQUAL (QUOTE (Z Y))
*20)))
12:(INSERTSET (QUOTE X)
(LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
*20)
T
(EQUAL (QUOTE (Z Y))
*20))))
by use of: INSERTSET
M2:(IF (EQUAL (LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
*20)
T
(EQUAL (QUOTE (Z Y))
*20)))
(NILSET))
(CONS(QUOTE X)
NIL)
(NOMINAL.OR
(CONSET (QUOTE X)
(LAMBDA *20 (IF (EQUAL (QUOTE (Y Z))
*20)
T
(EQUAL (QUOTE (Z Y))
*20))))
(LAMBDA *21
(EX *22 (AND (IS *22 (LAMBDA
*20
(IF (EQUAL (QUOTE (Y Z))
*20)
T
(EQUAL (QUOTE (Z Y))
*20))))
(IS *21 (CONSET(CAR *22)
(INSERTSET(QUOTE X)
(CDR *22))))))))
02:(LAMBDA *61
(IF (EQUAL *61 (QUOTE (X Y Z)))
T
(IF (EQUAL *61 (QUOTE (X Z Y)))
T
(IF (EQUAL *61 (QUOTE (Y X Z)))
T
(IF (EQUAL *61 (QUOTE (Y Z X)))
T
(IF (EQUAL *61 (QUOTE (Z X Y)))
T

```

```

                                (EQUAL *6I (QUOTE (Z Y X)))))))))
01:(LAMBDA *6I
  (IF (EQUAL *6I (QUOTE (X Y Z)))
    T
    (IF (EQUAL *6I (QUOTE (X Z Y)))
      T
      (IF (EQUAL *6I (QUOTE (Y X Z)))
        T
        (IF (EQUAL *6I (QUOTE (Y Z X)))
          T
          (IF (EQUAL *6I (QUOTE (Z X Y)))
            T
            (EQUAL *6I (QUOTE (Z Y X)))))))))

```

The result of recursive simplification is:

T
which is true. QED.

EX0=0/EX1=6/EX2=0/EX3=26/EX4=0/EX5=6/EX6=4/EX7=0/EX8=0/EX9=0/EX10=0/
ALL0=0/ALL1=0/ALL2=0/ALL3=1/ALL4=0/ALL5=0/ALL6=1/ALL7=0/ALL8=0/ALL9=0/ALL10=0/

COMPLEXITY PROJECT

ATP.COMPLEXITY contains four functions listed below:

(ANALYZE function.definition.being.analyzed basis.function.definition) ANALYZE tries to determine if the complexity of function definition being analyzed is linearly related to the given basis function. Later versions will handle multiple basis functions and nonlinear relationships. ANALYZE calls the routines COMFUN SIMFUN and the automatic theorem prover.

(COMFUN definition) Computes the complexity function definition of a given function definition.

(SIMFUN definition) Computes the simplified version of a function definition by deleting extraneous argument positions.

(BASISFUNS definition) Computes the immediate basis function definitions of a given complexity function definition.

The Complexity Analysis Project is concerned with the development of a reasoning system to automatically analyze and determine the complexity of computer programs. This research is important not only for theoretical computer science in providing a method for automating the process of analyzing the complexity of algorithms, but also for the practical problem of verifying time dependent properties of computer programs used in such real time areas as flight control systems. The current complexity analysis system called ANALYZE is capable of automatically analyzing the complexity of simple recursive

LISP functions. ANALYZE calls on our automatic deduction system SYMEVAL in a number of places in order to achieve its results. ANALYZE is described in section 1, and the use it makes of SYMEVAL is exemplified in section 2.

COMPLEXITY ANALYSIS SYSTEM: ANALYZE

We have developed a prototype system called ANALYZE for analyzing the complexity of recursive LISP functions. The basic approach to automatic program analysis used by ANALYZE is this: The user specifies a recursive LISP function F of which he wishes to analyze the complexity. The user may also specify that the analysis is to be performed in terms of certain basis functions which essentially compute the size of the input data of the original function. The system then does the following:

- (STEP 1) First, the COMplexity FUNction subsystem, called COMFUN, automatically produces a new LISP function $C.F$ which computes the complexity of F . This function is created by mimicking the recursive structure of F indicating the complexity of each branch.
- (STEP 2) Second, the SIMplification FUNction subsystem, called SIMFUN, tries to simplify the $C.F$ function by deleting irrelevant argument positions by SYMBOLically EVALuating the function body.
- (STEP 3) If they are not already specified, then the BASis FUNction subsystem, called BASFUN, automatically produces the possible appropriated basis functions. A basis function is a function which measures the size of some data object such as, for example, a tree.
- (STEP 4) Fourth, the system tries to guess a closed form solution to $C.F$ in terms of the basis functions.
- (STEP 5) Finally, using existential variables for coefficients, the system tries to prove that the recursive complexity function $C.F$ equals the conjectured closed form solution. In the course of the proof, the system may automatically determine explicit values for those existential variables.

The result of steps (1)-(5) is an algebraic formula expressing the complexity of F in terms of (a) the size of data object to which F is applied, and (b) the complexity of its subroutines. If the complexity of each subroutine and any subroutines called by such subroutines is determined by repeating steps (1)-(5), the complexity of F is then expressed as an algebraic formula containing only the complexity of primitive instructions and the size of the input data objects. The deductive parts of the system are based on the SYMEVAL theorem prover, the SYMMETRIC LOGIC, and the Real Algebra rule package. One novel aspect of this deduction system is that it integrates general structural inductive capabilities over arbitrary data objects along the lines of (Boyer1) with quantifier elimination techniques of the SYMMETRIC LOGIC and equation solving techniques of the real algebra theorem prover (Brown24). Some inductive parts of the system have been studied earlier in collaboration with Prof. Sten-Ake Tarnlund of Upsalla University (Brown5,10).

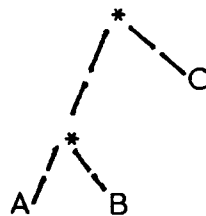
A SIMPLE EXAMPLE We suppose that the user of our proposed system wishes to analyze the complexity of the function FRINGE which computes the fringe of a binary tree T1 when L=NIL in terms of the number of nodes in the tree T1. The (FRINGE L NIL) of a binary tree is a list of its leaves. The definition of FRINGE is:

```
(EQUAL (FRINGE T1 L)
  (IF (LISTP T1)
      (FRINGE (CAR T1) (FRINGE (CDR T1) L))
      (CONS T1 L)))
```

The definition of the function NODEKNT which counts nodes in a tree is:

```
(EQUAL (NODEKNT T1)
  (IF (LISTP T1)
      (PLUS 1 (PLUS (NODEKNT (CAR T1)) (NODEKNT (CDR T1))))
      1))
```

The binary tree is assumed to be constructed from LISP CONSES. For example (CONS (CONS A B) C) represents the tree:



The LISP functions are all written in the SYMEVAL's logical language, which includes logical expressions, real numbers and recursive functions of pure LISP. We write (IF p x y) instead of the usual LISP conditional (COND(p x)(T y)).

We ask the system to try to analyze the complexity of FRINGE in terms of the function NODEKNT. The following is a trace of the complexity systems reasoning:

7_(ANALYZE FRINGE NODEKNT)

We are trying to determine whether the complexity of:

```
(EQUAL (FRINGE T1 L)
  (IF (LISTP T1)
      (FRINGE (CAR T1)
              (FRINGE (CDR T1)
                      L))
      (CONS T1 L)))
```

is related to the basis function:

```
(EQUAL (NODEKNT T1)
  (IF (LISTP T1)
      (PLUS 1 (PLUS (NODEKNT (CAR T1))
```

(NODEKNT (CDR T1))))

1))

(STEP 1) After the system states the problem, the subsystem COMFUN creates a recursive function which computes the abstract complexity of FRINGE: The local constant (A0007) is the complexity of taking the true branch of the LISTP test except for the complexity of the recursive calls to FRINGE which are mentioned explicitly. The local constant (A0008) is the complexity of taking the false branch of the test.

The complexity function of FRINGE is:

```
(EQUAL (C.FRINGE T1 L)
  (IF (LISTP T1)
    (PLUS (A0007)
      (PLUS (C.FRINGE (CAR T1)
        (FRINGE (CDR T1)
          L))
        (C.FRINGE (CDR T1)
          L))))
    (A0008)))
```

where the local constants are defined as follows in terms of the complexities of the primitive operations of LISP. For example (C.LISTP) is the complexity of executing the LISTP function and (C-VAR) is the complexity of looking up the value of a variable in a shallow binding environment.

```
((EQUAL (A0007)
  (PLUS (C.IF.T)
    (PLUS (C.LISTP)
      (PLUS (C-VAR)
        (PLUS (TIMES 2 (C-BIND))
          (PLUS (C.CAR)
            (PLUS (C-VAR)
              (PLUS (C.CDR)
                (PLUS (C-VAR)
                  (C-VAR))))))))))
  (EQUAL (A0008)
    (PLUS (C.IF.NIL)
      (PLUS (C.LISTP)
        (PLUS (C-VAR)
          (PLUS (TIMES 2 (C-BIND))
            (PLUS (C.CONS)
              (PLUS (C-VAR)
                (C-VAR))))))))))
```

(STEP 2) The subsystem SIMFUN now tries to simplify the complexity function definition just produced:

Observing that the variable L is not used in the body of the definition, it follows that the complexity function simplifies to the new complexity function:

```

(EQUAL (C.FRINGER T1)
  (IF (LISTP T1)
    (PLUS (A0007)
      (PLUS (C.FRINGER (CAR T1))
        (C.FRINGER (CDR T1))))
    (A0008)))

```

(STEP 3) Step three is omitted in this example because we already suggested to the system that NODEKNT was an appropriate basis function.

(STEP 4) An appropriate complexity conjecture relating C.FRINGER to NODEKNT is now produced:

You hinted that the complexity of FRINGER was related to the basis function: NODEKNT.

We will now try to see if its linearly related to that basis by first forming an expression stating that fact:

```

(ALL T1 (EQUAL (C.FRINGER T1)
  (PLUS (TIMES X (NODEKNT T1))
    Y)))

```

and then simplifying this expression as much as possible using our automatic theorem prover. The result returned by our theorem prover will be logically equivalent to this original expression.

(STEP 5) The conjectured relation between the complexity function and the basis is now proven:

The first SYMBolic EVALuation

The expression to be recursively simplified is:

```

(ALL T1 (EQUAL (PLUS (C.FRINGER T1)
  (PLUS (TIMES X (NODEKNT T1))
    Y)))

```

The result of recursive simplification is:

```

(ALL T1 (EQUAL (PLUS (C.FRINGER T1)
  (PLUS (MINUS (TIMES X (NODEKNT T1))
    (MINUS Y)))
  0))

```

Induction is now tried giving a new expression to simplify.

The expression to be recursively simplified is:

```

(AND
  (ALL T1 (IMPLIES (NOT (LISTP T1))
    (EQUAL (PLUS (C.FRINGER T1)

```

```

                                (PLUS (MINUS (TIMES X (NODEKNT T1)))
                                (MINUS Y)))
                                0)))
(ALL
  T1
  (IMPLIES
    (AND(LISTP T1)
      (AND(EQUAL (PLUS (C.FRINGER (CAR T1))
        (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
        (MINUS Y)))
        0)
      (EQUAL (PLUS (C.FRINGER (CDR T1))
        (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
        (MINUS Y)))
        0)))
    (EQUAL (PLUS (C.FRINGER T1)
      (PLUS (MINUS (TIMES X (NODEKNT T1)))
      (MINUS Y)))
      0))))

```

The result of recursive simplification is:

```

(IF (EQUAL (PLUS (A0008)
  (PLUS (MINUS X)
  (MINUS Y)))
  0)
  (EQUAL (PLUS (A0007)
    (PLUS Y (MINUS X)))
    0)
  NIL)

```

end of deduction

63798 conses

174.162 seconds

11.479 seconds, garbage collection time

We call the theorem prover again, this time letting it solve for the unknowns.

The expression to be recursively simplified is:

```

(IF (EQUAL (PLUS (A0008)
  (PLUS (MINUS X)
  (MINUS Y)))
  0)
  (EQUAL (PLUS (A0007)
    (PLUS Y (MINUS X)))
    0)
  NIL)

```

The result of recursive simplification is:


```

(IF (EQUAL (PLUS (A0008)
                 (PLUS (MINUS X)
                     (MINUS Y)))
  0)
  (EQUAL (PLUS (A0007)
              (PLUS (TIMES -2 X)
                  (A0008)))
  0)
  NIL)

```

end of deduction
 1412 conses
 2.685 seconds
 1.836 seconds, garbage collection time

Observing that (IF p x NIL) means ((AND p x) we see that the Automatic theorem prover has simplified the original closed form expression to an equivalent expression which is essentially a conjunction of linear equations which when solved give explicit values for the unknowns X and Y. By solving these two linear equations we see that:

$$\begin{aligned}
 X &= ((A0007) + (A0008)) / 2 \\
 Y &= ((A0008) - (A0007)) / 2
 \end{aligned}$$

where (A0007) and (A0008) are defined by the local definitions which in turn are defined in terms of the complexities of the primitive LISP operations.

Thus not only has the system proven the theorem:

```

(EX X (EX Y
  (ALL T1 (EQUAL (C.FRINGER T1)
                (PLUS (TIMES X (NODEKNT T1)) Y)))
  ))

```

where the unknowns X and Y are interpreted as being existentially quantified but in the course of proving this theorem, it has computed the only possible values for X and Y which make the expression true. Thus, in fact it proves the stronger theorem:

```

(ALL T1 (EQUAL (C.FRINGER T1)
              (PLUS (TIMES ((A0007) + (A0008))/2 (NODEKNT T1))
                  ((A0008) - (A0007))/2 (NODEKNT T1) )))

```

Since the deductive system itself can handle existential variables, this greatly eases the burden on the inductive step (4) of the proposed system since that step will not have to worry about guessing the exact coefficients of a conjecture of a closed form solution.

USING THE SYMBOLIC EVALUATOR: SYMEVAL

Although SYMEVAL is used by ANALYZE in a number of places, for example to simplify the bodies of function definitions, its major use is in step 5 where it is used to prove the

equivalence between the closed form solution and the original complexity function. It is therefore worthwhile looking at this reasoning step in more detail in order to describe SYMEVAL's current abilities. The outline of the proof given below is presented in the manner as one might trace the execution of a LISP program. Essentially an input expression labeled In: is given to SYMEVAL which by application of an axiom produces a middle expression labeled Mn: which is then recursively simplified producing an output expression labeled On:. The key point is that In: is logically equivalent in the given theory to the immediately following Mn: and also to the immediately following On:. The n refers to the current level of tracing. By specifying what symbols to trace, SYMEVAL can be asked to present its reasoning at different levels of detail. In the following proof only a few key symbols have been traced, and a number of less important steps have been eliminated by hand. Nothing, however, has been added except English text. This proof involves a number of basic deduction facilities including the nine listed below. The first use in this proof of each of these nine facilities is marked by the same number.

- (1) Methods for deciding when to replace definitions including recursive definitions by their body.
- (2) Rules for the algebraic simplification of expressions about real numbers.
- (3) A rule for Noetherian Induction over arbitrary recursively constructed data structures and recursive definitions.
- (4) Propositional Logic based on an IF_THEN_ELSE_ construct.
- (5) Rules of a Quantificational Logic based on the SYMMETRIC LOGIC of reducing the scope of quantifiers.
- (6) The ability to return useful information as answers to subgoals rather than having to return True or False.
- (7) The ability to solve equations for interesting expressions which can be substituted into other expressions so as to help solve the problem.
- (8) Axioms about recursive data structures.
- (9) Instantiation Rules for Quantificational Logic. Note that each induction hypothesis is eliminated by noting that it is equivalent to true assuming the linear equation produced by the base case.

The proof is now given:

The expression to be recursively simplified is:

```
(ALL T1 (EQUAL (C.FRINGER T1)
                (PLUS (TIMES X (NODEKNT T1))
                      Y)))
```

(1) SYMEVAL expands the definition of C.FRINGER and then changes its "mind".

```

I1:(C.FRINGER T1)
  by use of: C.FRINGER
M1:(IF (LISTP T1)
      (PLUS (A0007)
            (PLUS (C.FRINGER (CAR T1))
                  (C.FRINGER (CDR T1)))))
      (A0008))
O1:(C.FRINGER T1)

```

(2) The Real algebra equality rule is applied.

```

I1:(EQUAL (C.FRINGER T1)
          (PLUS (TIMES X (NODEKNT T1))
                Y))
  by use of: (LISPLINK REQUAL)
M1:(EQUAL (PLUS (C.FRINGER T1)
                (PLUS (MINUS (TIMES X (NODEKNT T1))
                        (MINUS Y))))
          0)
O1:(EQUAL (PLUS (C.FRINGER T1)
                (PLUS (MINUS (TIMES X (NODEKNT T1))
                        (MINUS Y))))
          0)

```

the result of recursive simplification is:

```

(ALL T1 (EQUAL (PLUS (C.FRINGER T1)
                    (PLUS (MINUS (TIMES X (NODEKNT T1))
                            (MINUS Y))))
              0))

```

(3) Induction is now tried giving a new expression to simplify:

```

(AND
  (ALL T1 (IMPLIES (NOT (LISTP T1))
                  (EQUAL (PLUS (C.FRINGER T1)
                              (PLUS (MINUS (TIMES X (NODEKNT T1))
                                      (MINUS Y))))
                        0)))
  (ALL T1
    (IMPLIES
      (AND (LISTP T1)
            (AND (EQUAL (PLUS (C.FRINGER (CAR T1))
                            (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))
                                    (MINUS Y))))
                        0)
                (EQUAL (PLUS (C.FRINGER (CDR T1))
                            (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))
                                    (MINUS Y))))
                        0)))
      (EQUAL (PLUS (C.FRINGER T1)
                    (PLUS (MINUS (TIMES X (NODEKNT T1))
                            (MINUS Y))))
            0)))

```

```

                (PLUS (MINUS (TIMES X (NODEKNT T1)))
                    (MINUS Y)))
            0))))

```

The Base Case of the Induction is Evaluated

```

I1:(IMPLIES (IF (LISTP T1)
                NIL T)
            (EQUAL (PLUS (C.FRINGER T1)
                        (PLUS (MINUS (TIMES X (NODEKNT T1)))
                            (MINUS Y))))
                0))
    by use of: IMPLIES
M1:(IF (IF (LISTP T1)
           NIL T)
      (EQUAL (PLUS (C.FRINGER T1)
                  (PLUS (MINUS (TIMES X (NODEKNT T1)))
                      (MINUS Y))))
          0)
    T)

```

(4) C.FRINGER becomes (A0008) in the Base Case

```

I3:(C.FRINGER T1)
    by use of: C.FRINGER
M3:(IF (LISTP T1)
      (PLUS (A0007)
            (PLUS (C.FRINGER (CAR T1))
                  (C.FRINGER (CDR T1)))))
    (A0008))
03:(A0008)
I3:(NODEKNT T1)
    by use of: NODEKNT
M3:(IF (LISTP T1)
      (PLUS 1 (PLUS (NODEKNT (CAR T1))
                    (NODEKNT (CDR T1)))))
    1)
03:1

```

The Base Case evaluated

```

01:(IF (LISTP T1)
      T
      (EQUAL (PLUS (A0008)
                  (PLUS (MINUS X)
                      (MINUS Y))))
          0))

```

(5) The quantifier is eliminated on the Base Case

```

11:(ALL T1 (IF (LISTP T1)
                T
                (EQUAL (PLUS (A0008)
                              (PLUS (MINUS X)
                                    (MINUS Y)))
                        0)))
  by use of: (LISPLINK SYMALL

```

```

M1:(IF (EQUAL (PLUS (A0008)
                    (PLUS (MINUS X)
                          (MINUS Y)))
           0)
        T
        (ALL T1 (IF (LISTP T1)
                      T NIL))))
01:(EQUAL (PLUS (A0008)
                (PLUS (MINUS X)
                      (MINUS Y)))
           0)

```

(6) The Remaining problem after evaluating the Base

```

11:(AND
  (EQUAL (PLUS (A0008)
                (PLUS (MINUS X)
                      (MINUS Y)))
           0)
  (ALL
   T1
   (IMPLIES
    (AND
     (LISTP T1)
     (AND (EQUAL (PLUS (C.FRANGE (CAR T1))
                       (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
                             (MINUS)))
            0)
          (EQUAL (PLUS (C.FRANGE (CDR T1))
                      (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                            (MINUS Y)))
                  0)))
    (EQUAL (PLUS (C.FRANGE T1)
                  (PLUS (MINUS (TIMES X (NODEKNT T1)))
                        (MINUS Y)))
            0))))

```

Evaluating the Induction Step

```

12:(IMPLIES
  (IF (LISTP T1)
       (IF (EQUAL (PLUS (C.FRANGE (CAR T1))

```

```

                                (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
                                (MINUS Y)))
                                0)
(EQUAL (PLUS (C.FRINGER (CDR T1))
              (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                    (MINUS Y))))
0)
NIL)
NIL)

(EQUAL (PLUS (C.FRINGER T1)
              (PLUS (MINUS (TIMES X (NODEKNT T1)))
                    (MINUS Y))))
0))
by use of: IMPLIES

```

C.FRINGER includes (A0007) on the Induction Step

```

I4:(C.FRINGER T1)
  by use of: C.FRINGER
M4:(IF (LISTP T1)
      (PLUS (A0007)
            (PLUS (C.FRINGER (CAR T1))
                  (C.FRINGER (CDR T1)))))
      (A0008)
04:(PLUS (A0007)
        (PLUS (C.FRINGER (CAR T1))
              (C.FRINGER (CDR T1)))))
I4:(NODEKNT T1)
  by use of: NODEKNT
M4:(IF (LISTP T1)
      (PLUS I (PLUS (NODEKNT (CAR T1))
                    (NODEKNT (CDR T1)))))
      I)
04:(PLUS I (PLUS (NODEKNT (CAR T1))
                 (NODEKNT (CDR T1)))))

```

(7) The hypothesis is solved for (C.FRINGER(CDR T1)) and substituted into the conclusion.

```

M4:(IF
  (EQUAL (PLUS (C.FRINGER (CDR T1))
                (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                      (MINUS Y))))
0)
(EQUAL
  (PLUS
    (A0007)
    (PLUS
      (C.FRINGER (CAR T1))
      (PLUS
        (PLUS Y (TIMES X (NODEKNT (CDR T1))))

```

```

      (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
            (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                  (MINUS Y))))))
    0)
  T)

```

which is then simplified

```

04:(IF
  (EQUAL (PLUS (C.FRINGER (CDR T1))
                (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                      (MINUS Y))))
    0)
  (EQUAL (PLUS (A0007)
                (PLUS (C.FRINGER (CAR T1))
                      (PLUS (MINUS X)
                            (MINUS (TIMES X (NODEKNT (CAR T1)))))))
    0)
  T)

```

The Hypothesis is solved for (C.FRINGER(CAR T1)) and then substituted into the conclusion

```

M4:(IF
  (EQUAL (PLUS (C.FRINGER (CAR T1))
                (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
                      (MINUS Y))))
    0)
  (IF
    (EQUAL (PLUS (C.FRINGER (CDR T1))
                  (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                        (MINUS Y))))
      0)
    (EQUAL (PLUS (A0007)
                  (PLUS (PLUS Y (TIMES X (NODEKNT (CAR T1))))
                        (PLUS (MINUS X)
                              (MINUS (TIMES X (NODEKNT
                                                    (CAR T1)))))))
      0)
    T)
  T)

```

which is then simplified

```

04:(IF
  (EQUAL (PLUS (C.FRINGER (CAR T1))
                (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
                      (MINUS Y))))
    0)
  (IF (EQUAL (PLUS (C.FRINGER (CDR T1))
                  (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                        (MINUS Y))))
    0)
  T)

```

```

      0)
    (EQUAL (PLUS (A0007)
      (PLUS Y (MINUS X)))
      0)
    T)
  T)

```

The Result of Evaluating the Induction Step

```

02:(IF
  (LISTP T1)
  (IF (EQUAL (PLUS (C.FRINGER (CAR T1))
    (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
    (MINUS Y)))
    0)
    (IF (EQUAL (PLUS (C.FRINGER (CDR T1))
      (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
      (MINUS Y)))
      0)
      (EQUAL (PLUS (A0007)
        (PLUS Y (MINUS X)))
        0)
      T)
    T)
  T)

```

(8) The (ALL T1) QUANTIFIER IS REDUCED IN SCOPE

```

12:(ALL
  T1
  (IF
    (LISTP T1)
    (IF (EQUAL (PLUS (C.FRINGER (CAR T1))
      (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
      (MINUS Y)))
      0)
      (IF (EQUAL (PLUS (C.FRINGER (CDR T1))
        (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
        (MINUS Y)))
        0)
        (EQUAL (PLUS (A0007)
          (PLUS Y (MINUS X)))
          0)
        T)
      T)
    T))
  by use of: (LISPLINK SYMALL)

```

resulting in

```

M2:(IF

```



```

(EQUAL (PLUS (A0007)
              (PLUS Y (MINUS X)))
0)
T
(ALL
  T1
  (IF
    (LISTP T1)
    (IF
      (EQUAL (PLUS (C.FRINGER (CAR T1))
                  (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
                      (MINUS Y)))
0)
      (IF (EQUAL (PLUS (C.FRINGER (CDR T1))
                      (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                          (MINUS Y)))
0)
          NIL T)
      T)
    T)))

```

The Quantified sub expression is examined

```

I3:(ALL
  T1
  (IF
    (LISTP T1)
    (IF
      (EQUAL (PLUS (C.FRINGER (CAR T1))
                  (PLUS (MINUS (TIMES X (NODEKNT (CAR T1))))
                      (MINUS Y)))
0)
      (IF (EQUAL (PLUS (C.FRINGER (CDR T1))
                      (PLUS (MINUS (TIMES X (NODEKNT (CDR T1))))
                          (MINUS Y)))
0)
          NIL T)
      T)
    T))

```

by use of: (LISPLINK SYMALL)

T1 is replaced by (CONS *1 *2)

```

M3:
(ALL
  *1
  (ALL
    *2
    (IF
      (LISTP (CONS *1 *2))
      (IF

```

```

(EQUAL
  (PLUS (C.FRINGER (CAR (CONS *1 *2)))
    (PLUS (MINUS (TIMES X
      (NODEKNT (CAR (CONS *1 *2))))))
    (MINUS Y)))
  0)
(IF
  (EQUAL
    (PLUS (C.FRINGER (CDR (CONS *1 *2)))
      (PLUS (MINUS (TIMES X
        (NODEKNT
          (CDR (CONS *1 *2))))))
      (MINUS Y)))
    0)
  NIL T)
T)
T)))

```

(9) Resulting in 03 below because

```

15:(ALL *2
  (IF (EQUAL (PLUS (C.FRINGER *2)
    (PLUS (MINUS (TIMES X (NODEKNT *2)))
      (MINUS Y)))
    0)
  NIL T))

```

by use of: EX

```

05:NIL
14:(ALL *1 (IF (EQUAL (PLUS (C.FRINGER *1)
  (PLUS (MINUS (TIMES X (NODEKNT *1)))
    (MINUS Y)))
    0)
  NIL T))

```

by use of: EX

```

04:NIL
03:NIL

```

The Result of the Induction Step

```

02:(EQUAL (PLUS (A0007)
  (PLUS Y (MINUS X)))
  0)

```

the result of recursive simplification is:

```

(IF (EQUAL (PLUS (A0008)
  (PLUS (MINUS X)
    (MINUS Y)))
  0)
  (EQUAL (PLUS (A0007)

```

(PLUS Y (MINUS X)))
0)
NIL)

end of deduction
63798 conses
174.162 seconds
11.479 seconds, garbage collection time

EX1=0/EX2=0/EX3=0/EX4=0/EX5=0/EX6=0/EX7=0/EX8=0/EX9=0/EX10=0/
ALL1=0/ALL2=0/ALL3=0/ALL4=1/ALL5=1/ALL6=0/ALL7=2/ALL8=2/ALL9=1/ALL10=1/

CONCLUSION

We have constructed an entire automatic deduction and induction system based on a single principle which we call The Fundamental Deduction Principle. Unlike most other automatic deduction systems, this system does no unification whatsoever. Instead, it is based on the SYMMETRIC LOGIC technique of reducing the scope of quantifiers. We have used this system both as a programming language interpreter to make deductions in natural language theory and Ontology and as a more general deduction system to prove theorems and analyze the complexity of LISP functions. This research has not been done in a vacuum. Indeed, Bledsoe (Bledsoe2) argued over a decade ago that automatic deduction systems must INCLUDE ideas akin to our Fundamental Deduction Principle as opposed to the then prevailing Resolution viewpoint of deduction as being a problem of exploring a search space. However, the thesis of this research is incredibly stronger, for we are arguing that for many interesting theories in mathematics and computer science, that the Fundamental Deduction Principle is the ONLY idea that needs to be included. We were lead to this principle partly by trial and error in constructing deduction systems and suffering the effects of redundant expressions whenever we departed from this principle, and partly by Meltzer's contention that induction must in some way be related to deduction (Meltzer). The research of Boyer and Moore (Boyer) has also influenced us, in fact we have applied their techniques for Noetherian Induction in quantifier free logic to our Quantified logic. This research has also been influenced by the idea that computation is a very special case of deduction, and that a deductive system must be capable of computation. This is related to Kowalski's (Kowalski) thesis that deductive systems are capable of computation. For logical languages based on quantifier free functional representation, such as pure LISP, it is easy to achieve computational ability while obeying the fundamental deduction principle. However, for logical languages which include quantifiers and relational notation, achieving computational ability while obeying this principle would seem to be rather difficult since neither resolution nor unification generally obey this principle. Nevertheless we have succeeded in constructing a new deductive method called the SYMMETRIC LOGIC which has significant computational ability and which satisfies the fundamental deduction principle. This deductive method was first debugged in collaboration with Schwind (Brown14,15,17) by hand simulation of part of her theory (Schwind) for translating natural language into logic.

I wish to thank my students who have worked on this project, particularly Nelson Bishop who tested much of Schwind's grammar and Song Park who worked on the set theory examples.

REFERENCES

1. Bibel2, W., and Schreiber, J. "Proof Search in a Gentzen like system of first order logic", INTERNATIONAL COMPUTING SYMPOSIUM 1975, North Holland, Amsterdam.
2. Bledsoe1, W. W. "Splitting and Reduction Heuristics in Automatic Theorem Proving" ARTIFICIAL INTELLIGENCE Vol. 2, No. 1, Spring 1971.
3. Bledsoe2, W. W. "Non Resolution Theorem Proving" ARTIFICIAL INTELLIGENCE, Vol. 9, 1977.
4. Boyer1, Robert S., and Moore J. Strother, A COMPUTATIONAL LOGIC, Academic Press 1979.
5. Brown1, F. M., "A Deductive System for Elementary Arithmetic", 2nd AISB CONFERENCE PROCEEDINGS, Edinburgh, July 1976.
6. Brown3, F. M., "Doing Arithmetic Without Diagrams", ARTIFICIAL INTELLIGENCE, Vol. 8, Spring 1977.
7. Brown4, F. M., "A Theorem Prover for Elementary Set Theory", 5th INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, MIT, August 1977. Also the abstract is in the WORKSHOP ON AUTOMATIC DEDUCTION COLLECTED ABSTRACTS, MIT, August 1977.
8. Brown5, F. M., and Sten-Ake Tarnlund, "Inductive Reasoning in Mathematics", 5th INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, MIT, August 1977.
9. Brown6, F. M., "Towards the Automation of Set Theory and its Logic", ARTIFICIAL INTELLIGENCE, Vol. 10, 1978.
10. Brown10, F. M., and Sten-Ake Tarnlund, "Inductive Reasoning on Recursive Equations", ARTIFICIAL INTELLIGENCE, Vol. 12 #3, November 1979.
11. Brown11, F. M., "An investigation into the goals of research in Automatic Theorem Proving as related to Mathematical Reasoning" ARTIFICIAL INTELLIGENCE, 1980.
12. Brown12, F. M., "A Sequent Calculus for Modal Quantificational Logic", 3rd AISB/GI CONFERENCE PROCEEDINGS, Hamburg, July 1978.
13. Brown13, F. M., and C. B. Schwind, "Analyzing and Representing Natural Language in Logic", 3rd AISB/GI CONFERENCE PROCEEDINGS, Hamburg, July 1978.

14. Brown14, F. M., and C. B. Schwind, "Towards an Integrated Theory of Natural Language Understanding", To appear on microfilm in Linguistics Journal. Also extended abstract in 4th COLING CONFERENCE PROCEEDINGS, Bergen, 1978.
15. Brown15, F. M., and C. B. Schwind, "Outline of an Integrated Theory of Natural Language Understanding", REPRESENTATION AND PROCESSING OF NATURAL LANGUAGE, ed. Leonard Bolc, Carl Hanser Verlag 1980.
16. Brown17, F. M., "An Automatic Proof of the Completeness of Quantificational Logic", Department of Artificial Intelligence Research Report 52, 1978.
17. Brown18, F. M., "A Theorem Prover for Metatheory", 4th CONFERENCE ON AUTOMATIC THEOREM PROVING, Austin, Texas, 1979.
18. Brown20, F. M., "Logic Algorithms for Natural Numbers", TR 108, 1979.
19. Brown21, F. M., "A Deductive System for Meta Theoretic Reasoning", TR 132, January 1980.
20. Brown24, F. M., "A Deductive System for Real Algebra", TR 141, March 1980.
21. Brown25, F. M., "A Sequent Calculus for Intensional Logic", TR 143, April 1980.
22. Brown 26, F. M., "Computational with Automatic Theorem Provers", to appear in the proceedings of the NSF workshop on Logic Programming, Los Angeles, 1981.
23. Chester1, Daniel, "HCPRVR: An Interpreter for Logic Programs" proceedings of THE FIRST ANNUAL NATIONAL CONFERENCE on ARTIFICIAL INTELLIGENCE. The American Association for Artificial Intelligence. 1980.
24. Colmerauer, A., Kanoui H., Van Caneghem, M. "Last Steps Towards an Ultimate Prolog", PROCEEDINGS OF THE SEVENTH JOINT INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, pages 947-948, University of British Columbia, 1981.
25. Frege Gottlob, 'Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought' 1879, in FROM FREGE TO GODEL, Harvard Univ. Press 1967.
26. Henry, D. P., MEDIEVAL LOGIC AND METAPHYSICS, Hutchinson & CO LTD, 1972.
27. Kowalski, Robert, "Predicate Logic as programming language", PROCEEDINGS IFIP, 1974.
28. Lushei, E. C., THE LOGICAL SYSTEMS OF LESNIEWSKI, 1962.
29. McCarthy, J., et al., LISP 1.5 Programmer's Manual, MIT Press, 1966.
30. Meltzer, B. "The Programming of Deduction and Induction", DCL memo 45. Department of Computational Logic, The University of Edinburgh, 1971.

31. Moore, J., "Computational Logic: Structure Sharing and Proof of Program Properties", PhD thesis, University of Edinburgh, 1973.
32. Pastre D., DEMONSTRATION AUTOMATIQUE DE THEOREMS EN THEORIE DES ENSEMBLES, These de 3eme cycle, Paris VI, 1976.
33. Prawitz, D., "An Improved Proof Procedure", THEORIA 26, 1960.
34. Robinson, J. A., "A machine oriented logic based on the resolution principle", J.ACM 12, 1965.
35. Schwind, C. B., EIN FORMALISMUS ZUR BESCHREIBUNG DER SYNTAX UND BEDEUTUNG VON FRAGE-ANTWORT-SYSTEMEN, Ph.D. thesis, Technischen Universitat, Munchen, 1977.
36. Simmons1, R. F., and Chester, D. L., "Relating Sentences and Semantic Networks with Procedural Logic", CACM (to appear) 1979.
37. Simmons2, R. F., "A Narrative Schema in Procedural Logic" in Clark, K., and Tarnlund, S. (eds.) LOGIC PROGRAMMING (in press), Academic Press, New York, 1982.
38. Szabo, M. E., ed., THE COLLECTED PAPERS OF GERHARD GENTZEN, North Holland, Amsterdam, 1969.
39. Wang2, Hao, "On the long-range prospects of automatic theorem-proving", LECTURE NOTES IN MATHEMATICS: SYMPOSIUM ON AUTOMATIC DEMONSTRATION, held at Versailles/France 1968, Springer-Verlag 1970.

APPENDIX A
PROGRAM AGENDA

PROGRAM AGENDA

Wednesday, 20 April 1983

- 0730 **Badging and Check-in.** Main Auditorium, Naval Surface Weapons Center, White Oak, Silver Springs, Maryland
- 0830 **Welcome.** Capt. J. E. Fernandes, Commander, Naval Surface Weapon Center (NSWC)

EXECUTIVE SESSION

Chairman, Executive Session. Dr. Robert B. Oswald, Technical Director, US Army Electronics Research and Development Command (ERADCOM)

- 0835 **Opening Remarks.** Dr. Robert Oswald, HQ ERADCOM.
- 0840 **Introduction.** Dr. Robert Weigle, Director, US Army Research Office (ARO)
- 0850 **Overview.** Dr. Marvin E. Lasser, Director of Army Research, Office of the Deputy Chief of Staff for Research Development and Acquisition.
- 0905 **Opening Remarks.** BG Alan B. Salisbury, US Army, Director, Special Task Force, Joint Tactical Fusion Program.
- 0920 **Keynote Address: Artificial Intelligence (AI)/Robotics.** Dr. Edith W. Martin, Deputy Undersecretary of Defense for Research and Engineering (Advanced Technology)
- 0950 **Break**

SESSION I

GENERAL

Chairman Session I. Dr. Jagdish Chandra, Director, Mathematical Sciences Division, US Army Research Office (ARO).

- 1010 **Conference Highlights.** Dr. Jagdish Chandra, Director, Mathematical Sciences Division, ARO.
- 1020 **Requirements for Common Sense Knowledge in Artificial Intelligence.** Professor John McCarthy, Dept. of Computer Science, Stanford University.

SESSION II

DEPARTMENT OF DEFENSE PROGRAM REVIEW

Chairman Session II. Dr. Berthold Zarwyn, Science Advisor to the Technical Director, HQ, ERADCOM. "A Review of DOD and Other Service AI Programs."

- 1110 **Introduction.** Dr. Berthold Zarwyn, HQ ERADCOM.
- 1120 **DARPA Research Programs in AI.** Commander Ronald Ohlander, USN, Information Processing Techniques Office, Defense Advanced Research Projects Agency.
- 1155 **Navy AI Programs.** Dr. Jude E. Franklin, US Naval Research Laboratory.
- 1230 **Lunch**
- 1340 **Air Force AI Programs.** MAJ William Price, USAF, Air Force Office of Scientific Research.

SESSION III

IMAGE UNDERSTANDING

Chairman, Session III. Mr. G. David Singer, US Army Night Vision and Electro-Optics Laboratory. "Applications of AI to Image Understanding Such as Target Detection and Classification, Map Analysis, Photo-interpretation, and Autonomous Navigation (Needs, Problems with Current Methods and AI as a Solution)."

- 1415 **Introduction.** Mr. G. David Singer, Night Vision and Electro-Optics Laboratory.
- 1430 **Expert Systems for Terrain Analysis.** Dr. Robert Leighty, US Army Engineer Topographic Laboratory.
- 1500 **Topographic Primal Techniques.** Professor Robert Haralick, Department of Electrical Engineering, Virginia Polytechnic Institute.
- 1530 **Integrated Target Classification.** Dr. Visvaldis A. Vitols, Rockwell International.
- 1555 **BREAK**
- 1610 **Use of Expert Systems in Image Understanding.** Professor Laveen Kanal, Director of Laboratory for Pattern Analysis, University of Maryland.

SESSION III (Continued)

- 1640 **AI Context Analysis for Automatic Target Recognition.** Dr. Andrew J. Spiessbach, Project Engineer, and Mr. John F. Gilmore, Martin Marietta Aerospace.
- 1705 **The Use of AI in Target Classification.** Dr. Tod Levitt, and Dr. Raj Aggarwal, Honeywell Systems Research Center.
- 1730 **END SESSION III**
- 1800-2000 **Social Mixer** at Holiday Inn, Silver Spring

Thursday, 21 April 1983

SESSION IV

INTELLIGENCE FUSION

Chairman Session IV. Dr. Gerald R. Andersen, US Army Materiel Development and Readiness Command. "Actual and Potential Applications of AI to Intelligence Fusion."

- 0830 **Introductory Remarks.** Dr. Gerald R. Anderson, HQ DARCOM.
- 0835 **Men and Machines in Tactical Intelligence.** Dr. Edward Taylor, Director of Requirements and Analysis, Defense Systems Group, TRW, Inc.
- 0910 **Attempts at Applying AI to Situation Analysis.** Dr. Carl Verhey, Scientific Advisor, US Army Intelligence School and Center.
- 0945 **Application of Artificial Intelligence to Tactical Operations—Corps.** Colonel Don Gordon, and Major Timothy A. Campen, HQ Joint Special Operations Command.
- 1020 **BREAK**
- 1035 **An AI Approach to Multisensor Target Identification.** (Secret Presentation) Mr. Roland Payne, Vice President, Advanced Information and Defense Systems.
- 1110 **ANALYST: An Expert System for Sensor Processing.** Mr. Peter Bonasso, MITRE Corporation.
- 1120 **An Architecture for Application of AI Techniques to the Threat Warning Function.** Dr. Deane F. Babcock, Systems Techniques Laboratory, SRI.
- 1155 **Software Tools for Intelligence Fusion.** Professor Andrew B. Whinston, Kannert Graduate School of Management, Purdue University.
- 1230 **LUNCH**

SESSION V

SIGNAL PROCESSING

Chairman Session V. Mr. Douglas Chubb, Computer Scientist, Signals Warfare Laboratory.
"Communications and Noncommunications AI Signal Processing Systems."

- 1400 **An Overview of the Applicability and Use of AI Techniques to the Processing of Communications and Noncommunications Signals.** Mr. Douglas Chubb, SWL. (SECRET Presentation)
- 1415 **A Message Understanding Front End for a Knowledge-Based Threat Warning System.** Dr. Christine A. Montgomery, Chief Scientist, Operating Systems Division, LOGICON. (NOFORN Presentation)
- 1450 **Radar Signal Processing.** Dr. Mark R. Nixon, TRW, Inc. (SECRET Presentation)
- 1525 **Tactical Indications and Warnings Analysis.** Professor Douglas Lenat, Department of Computer Science, Stanford University; Dr. Garo Kiremidjian, Senior Staff Scientist, ESL/TRW; and Mr. Albert Clarkson, Program Manager, ESL/TRW. (SECRET Presentation)
- 1600 **BREAK**
- 1615 **Syntax Problems with Speech Recognition in Simulator Training Systems.** Mr. Thomas Cutler, Marketing Research Manager, VERBEX.
- 1650 **Voice Interactive Computer Systems.** Professor Alan Biermann, Department of Computer Science, Duke University.
- 1730 **END SESSION V**

SESSION VI (Sheraton Inn)

ROBOTICS

Chairman Session VI. Dr. Frank D. Verderame, Assistant Director of Army Research, Office of the Deputy Chief of Staff for Research Development and Acquisition.

- 1930 **Spatial Reasoning for Mobility and Manipulation.** Professor Rodney Brooks, Research Scientist, Artificial Intelligence Laboratory, MIT.
- 2005 **Technological Assessment of Future Battlefield Robotic Applications.** Mr. Barry Brownstein, Manager, Digital Systems and Technology Section, and John Reidy, Principal Research Engineer, Equipment Development Section, Battelle Columbus Laboratories.

SESSION VI (Continued)

- 2040 **AI Applications for Autonomous Vehicle Development.** Dr. David Y. Tseng, Section Head, and Bruce L. Bulloch, Computer Science and Image Analysis Section, Hughes Research Laboratories.
- 2115 **End Session VI**

Friday, 22 April 1983

SESSION VII

GENERAL APPLICATIONS OF EXPERT SYSTEMS

Chairman Session VII. LTC Henry Langendorf, USA, Robotics and Artificial Intelligence Office, US Army Soldiers Support Center.

"Knowledge Based Expert Systems and their Construction, with Examples of Training and Tactical Decision Making Applications."

- 0800 **Introduction.** LTC Henry Langendorf, US Army Soldier Support Center.
- 0815 **Overview of Expert Systems.** Dr. William Gevarter, Office of Aeronautics and Space Technology, NASA.
- 0900 **Applications of Knowledge Engineering.** Ms. H. Penny Nii, Department of Computer Sciences, Stanford University.
- 0945 **BREAK**
- 1000 **Knowledge Acquisition and Evaluation.** Professor Donald Loveland, Department of Computer Science, Duke University.
- 1045 **Knowledge Based Systems in Training.** Dr. Albert Stevens, Information Sciences Division, BBN.
- 1130 **Artificial Intelligence, an Expert Consultant System and an Intelligence Control Strategy.** Dr. James Slagle, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratories.
- 1230 **LUNCH**

SESSION VIII

NATURAL LANGUAGES/AUTOMATIC PROGRAMMING

Chairman Session VIII. Dr. Stephen Wolff, Ballistics Research Laboratory.

- 1330 **Introduction.** Dr. Stephen Wolff, Ballistics Research Laboratory.
- 1340 **What Systems Have to Say.** Professor Bonnie Lynn Webber, Department of Computer and Information Science, University of Pennsylvania.
- 1420 **Experimental Logic and Analysis of Computer Programs.** Professor Frank Brown, Department of Computer Sciences, University of Texas at Austin.
- 1500 **Break**
- 1515 **Research Directions in Automatic Programming.** Professor Elaine Kant, Computer Science Department, Carnegie-Mellon University.
- End Session VIII**
- 1555 **Closing Remarks,** Dr. Jagdish Chandra, ARO.

APPENDIX B
CONFERENCE ATTENDEES

CONFERENCE ATTENDEES

Abrams, Harry C.
USAFAS

Aggarwal, Raj K.
Honeywell Inc.

Ahern, Chuck J.
General Dynamics

Ahlers, Robert H., Jr.
Naval Training Equipment Ctr

Ahn, Byong H.
US Army ERADCOM

Alfano, Joseph
Naval Air Test Center

Ames, Henry S.
Lawrence Livermore Nat'l Lab

Amory, Robert

Andes, David K.
Naval Weapons Center

Antony, Richard T.
Harry Diamond Lab

Arbabi, Hansur
IBM

Aronson, Alan R.
INCO, Inc.

Atkinson, Gerald L.
Analytic Sciences Corp.

Babcock, Dean F.
SRI International

Babcock, Scott M.
Oak Ridge National Lab

Bailey, Timothy J.
Army Comb. Arms Opns Res

Balicki, Frederick W.
Harry Diamond Lab

Batz, Joseph C.
OSD

Baumgardt, Douglas
ENSCO, Inc.

Baxter, Alan M.
GA Technologies Inc.

Beltracchi, Leo
Nuclear Regulatory Commission

Bening, Dale
Ford Aerospace & Comm. Corp.

Bennett, C. Leonard
Sperry Research Center

Benoit, John W.
MITRE Corporation

Benokraitis, Vitalius J.
AAI Corporation

Benton, John R.
USAETL

Bierman, Alan W.
Duke University

Bisbee, John
ITEK Optical Systems

Biswas, Gautam
Univ. of South Carolina

Black, Alan J.
AFRRI/CSC

Blair, Alan H.
BDM Corp.

Brown, Nathan H., Jr.

Bonasso, Russell P.
MITRE Corp.

Bonnell, Ronald D.
Univ. of South Carolina

Bowles, Ronald

Braudaway, Wesley K.
IBM

Braude, Eric J.
RCA AdvancedTech Lab

Brazil, David J.
SYSCON Corp.

Bregar, William S.
Univ. of Delaware

Briggs, Arthur B., Jr.
Texas Instruments

Bright, Carlisle R.
Ford Aerospace

Brockstein, Allan J.
Litton Guidance & Ctrl

Brodnax, Charles T.
E-Systems, Inc.

Brooks, Rodney A.
MIT AI Lab

Broome, Paul H.
Army Ballistic Res Lab

Brown, Frank M.
Univ. of Texas at Austin

Brown, Glenn A.
T&E International Inc.

Bohan, David
Systems Engineering

Brownstein, Barry J.
Battelle Columbus Labs

Busdiecker, Roy F.
Joint Tactical Fusion PMO

Cammarata, Ronald W.
Army Human Engrg Lab

Campen, Timothy A.
HQ JSOC J2-ADP

Cannon, Robert L.
Univ. of South Carolina

Cantrell, Ben H.
Naval Research Lab.

Carion, Felipe
Ford Aerospace & Comm. Corp. AMSAA

Carpenter, William A.
INCO, Inc.

Carroll, Roberta
USAETL-RI-CAI

Chandra, Jagdish
US Army Research Office

Chang, Andrew K.
FMC Corp.

Chang, Raykun Rosa
AI/Robotics

Chatlynne, Charles J.
Army ERADCOM

Cheh, May L.
National Library of Medicine

Chow, Sen-Te
NV&EOL

Choy, Steven J.
Harry Diamond Lab

Christman, Arthur C., Jr.
Harry Diamond Lab

Chu, Sai-Cheong A.
Bendix Corp.

Chubb, Douglas W. J.
US Army Signals Warfare Lab

Claffy, R. M.

Clarke, J. Dallas IV
DARPA

Clarkson, Albert G.
ESL, Inc.

Cohen, Edgar A., Jr.
NSWC

Cohen, Herbert E.

Coleman, Norman P.
ARRADCOM

Collins, Dean R.
Texas Instruments Inc.

Collins, Jack
Magnavox Data Systems, Inc.

Conger, C. Richard

Conly, James

Conrad, John C., Jr.
GE Space Systems Div

Cook, Thomas M.
Naval Ocean Systems Center

Cook, Dennis R.
Harry Diamond Labs

Corn, Philip B.
HQ AFSC/DLZ

Costanza, John
Harry Diamond Labs

Cox, Robert G.
SDC-Burroughs

Crombie, Michael A.
Army Engineer Topo Labs

Crone, Michael S.
HRB-Singer

Cronin, Terence M.
Army Signals Warfare Lab

Cross, George R.
Louisiana State Univ.

Cummings, Clifford I.
Xerox Electro-Optical Sys.

Curl, Clarence L.
Army (TRASANA)

Cutler, Thomas P.
Verbex/Exxon Enterprises

Cutting, John E.
Battelle Columbus Labs.

Dammond, John T.

Davis, John
NSA

Davis, Larry S.
ImTech, Inc.

Dean, L. Byron
Sandia National Labs

Deffenbaugh, Floyd D.
TRW

DeFoe, Douglas N.
CAI

deHaan, Henry J.
Army Research Institute

DeLauter, Joseph H.
HRB-Singer, Inc.

DeMeyer, Donald L. Lear Siegler Inc.	Fedorowycz, Bohdan W. General Dynamics	Galbavy, Alan G. PM TRADE
DeYoung, Tice F. Army Engineer Topo Labs	Feeney, Gerald F. USA INSCOM	Gale, Manfred MITRE Corporation
Diakides, Nicholas A. NV&EOL	Felder, Wilson N. TRW Defense Systems Group	Gambino, Lawrence A. Army Engineer Topo Labs
DiCarlo, Samuel S. Rome Air Development Center	Firschein, O. Lockheed Palo Alto Research Lab	Gandolfo, D. A. RCA
Dietz, David C. Systems Research Labs	Fisher, Matthew J. Avionics R&D Activity	Garry, Kent USAFAS
DiPaola, Robert A. Univ. of New York	Fitch, Robert J. E-Systems, Garland Division	Garvey, Deborah L. USAICS
Doherty, James	Foote, Alvin L. FMC, Ordnance Div Engineering	Gates, Kermit H. PAR Technology Corp.
Dorrough, Douglas C. Ford Aerospace & Comm. Corp.	Fox, Frederick W. Army Soldier Support Center	Gault, James W. Army Research Office
Driskel, Carl	Frank, Lawrence J. AIRMICS	Geffert, Terry L. SNIDER Engineering
Duncan, James R. US Navy Research Center	Franklin, Jude E. Naval Research Laboratory	Geesey, Roger A. BDM Corp.
Duncan, Roger MITRE Corp.	Franks, Edwin	Gevarter, William B. NASA Headquarters
Ealy, William	Frawley, William J. Knowledge Based Systems	Giddings, Nancy M. Honeywell, Inc.
Eckels, James R.	Freedman, Roy S. Hazeltine Corp.	Girard, Paul ONR
Edwards, Daniel L. USAETL	Frydman, Abraham	Glick, Norman S. National Security Agency
Egtert, Russell J. Ultrasystems, Inc.	Fu, K. S. Purdue University	Godfrey, Leon D. Army Comb. Arms Opns Res
Eppler, W. G. Lockheed Palo Alto Res. Lab	Funke, Maurice F. The BMD Corp.	Goehrig, George
Evans, Timothy D. US Army Research Office	Gaebel, Darrell	Goguen, Joseph A. SRI
Fahey, Andrew P. CAI	Gaev, Jonathan C ³ I Fusion	Goodhart, Curtis L. Naval Ocean Sys Center

Goodman, Harvey S. TRW DSG	Hayner, Robert E. USACDEC	Huntsberger, Terrance L. Univ. of South Carolina
Gordon, Donald Joint Special Opns Cmd	Healey, Paul R. American University	Iaquinto, Joseph F. HRB-Singer Inc.
Gormally, John	Hefley, William E. Lockheed Missiles & Space Co.	Ingersoll, Philip F.
Graham, Robert V. Hughes Aircraft Co.	Heigl, James J., Jr. SAI	Ingram, John C. Army Signals Warfare Lab
Gray, Frederic C. Defense Mapping School	Hendricks, Walter J. INCO, Inc.	Inselmann, Edmund H. Army Combined Arms Opns Res
Greene, John S., Jr. Martin Marietta Aerospace	Hill, Connie Ray Johnson Control, Inc.	Ireland, Terrance
Griese, William F. Hazeltine Corp.	Hinchion, Frank S. Martin Marietta Aerospace	James, J. N. Jet Propulsion Lab
Griffin, Arthur F. Hughes Aircraft Co.	Hoefke, Robert	Jaszlics, Ivan J. Martin Marietta Denver
Gross, Frederick E. SYSCON Corporation	Holly, Franklin	Jazwinski, Andrew H. Analytic Sciences Corp.
Gurman, Bernard S. AVRADCOM	Holmes, Willard M. US Army Missile Command	Jenkins, Randy Magnavox Data Systems, Inc.
Hagan, Lucy B. HQ DARCOM	Horton, David A. Westinghouse Electric Defense Ctr	Jenkins, Robert S. HQ DARCOM
Hahn, Samuel S. ESL	Howell, Elizabeth E. SAI	Johnson, John US Army ERADCOM
Hall, Gregory V. US Army Military Police Sch.	Howell, Hardy M. US Army Medical R&D Command	Jones, Edward C. NV&EOL
Hallnor, George	Howells, Frederick W. Systems Research Labs, Inc.	Jones, George R. NV&EOL
Haralick, Robert VA Polytechnic Institute	Huhns, Michael Neil Univ. of South Carolina	Jones, James T. Purdue University
Harper, Charles Magnavox Data Systems, Inc.	Hull, Kent S. Office of Naval Research	Jones Robert E., Jr. ODCSPER
Harrison, Kermit C. TRW Defense Systems Group	Hunt, Kenneth Ofc of Armor Force Mgt & Stand.	Jones, Terry L. NV&EOL

AD-A139 685

PROCEEDINGS OF THE ARMY CONFERENCE ON APPLICATION OF
ARTIFICIAL INTELLIGENCE (U) BATTELLE WASHINGTON
OPERATIONS DC B J TULLINGTON 31 JAN 84

4/4

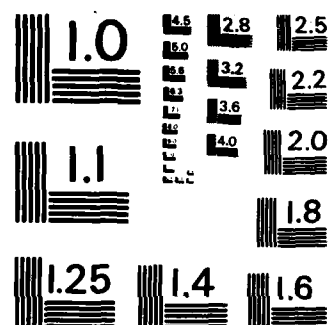
UNCLASSIFIED

DAG29-81-D-0100

F/G 5/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Juhn, Heinrich GCA Industrial Systems Group	Kitchell, Thomas J. Army Concepts Analysis Agency	Lemmer, John F. PAR Technology Corp.
Kahn, David A. MITRE Corporation	Kobler, Virginia P. US Army BMDATC	Lewandowski, Richard Emerson Electric
Kanal, Laveen N. University of Maryland	Kotin, Leon Communications-Electronics Cmd	Lewis, Virgil
Kant, Elaine Carnegie-Mellon University	Kovel, Steven M. Harry Diamond Lab	Lindsey, Elbert R. PRC/GIS
Kaplan, Walter NSWC	Krasner, Herb C. Harris Government Systems Sector	Llinas, James HRB-Singer
Karas, Leslie Battelle Columbus Labs	Kreinik, Herbert PRC/GIS	Loatman, R. Bruce PRC
Karp, Ira L. Advanced System Division	Krovetz, Robert J. Knowledge-Based Info Retriever	Love, Daniel L. NSWC
Katter, Robert V. LOGICON, Inc.	Lackey, Cora D. PRC/GIS	Loveland, Donald W. Duke University
Kaufman, Bradford A. Naval Surface Weapons Center	Langendorf, Henry S. AI and Robotics Office	Lubbers, David H. HRB-Singer, Inc.
Kelly, Michael D. BMD Corp.	Langley, Maureen B. Univ. of Wisconsin/River Falls	Lucero, Antonio B. Singal Processing Lab
Kelly, William C. US Army Missile Command	Lanphear, David K. INCO, Inc.	Lukes, George E. USAETL Research Institute
Kenig, Neil S. RCA Corporation	Lasher, William K. E-Systems	Lum, Rosalyn K. L. ITT-Advanced Tech Center
Kessler, H. M.	Lasser, Marvin E. ODCSRDA	Mack, Harold Ford Aerospace
Kim, John C. TRW Defense Systems Group	Law, Harold Y. H. AVRADCOM	Mackay, Lester NV&EOL
King, Jon J. Army Topographic Lab	Lawson, Walter R. NV&EOL	MacPherson, Douglas H. Army Res for Behav/Soc Sci.
Kiremidjian, Garo K. ESL, Inc.	Ledford, John F.	Maimone, Emanuel
Kirkpatrick, Lane GCA Industrial Systems Group	Leighty, Robert D. Army Engineer Topographic Labs	Manby, Mary F. HQ DARCOM

Mandelberg, Martin General Dynamics	McFarling, Leslie H. Litton Industries	Morefield, Charles L. VERAC, Inc.
Marinuzzi, John G. Los Alamos National Lab	McGehee, Eva L. Xerox Electro-Optical Systems	Morris, Derek
Markosian, Lawrence Z. Systems Control Technology	McGurk, Frank US Army War College	Morrissett, Grayson G. Army Logistics Center
Marner, Gene R. Army Aviation R&D Command	McKean, Richard E. Army Field Artillery School	Naef, Frederick E. Marketing
Marshall, Neal	McKenna, Robert J. Army Infantry School	Nagle, John B. Ford Aerospace & Comm. Corp.
Martin, Albert R. Harris Corporation - GISD	McKindra, Clayton Naval Surface Weapons Center	Narva, Marshall A. Army Research Institute
Martin, Nancy WANG Inst. of Grad. Studies	Mercier, Joseph L. Program Development	Nelson, Walter H. INCO, Inc.
Masters, Michael W. Naval Surface Weapons Center	Meyer, Stephen A. Army Aviation R&D Command	Nii, H. Penny Stanford University
Mastronianni, G. USAARL	Meyrowitz, Alan L. Office of Naval Research	Nixon, Mark R. TRW
Matyskiela, Walter W. TRW Defense Systems Group	Michel, Rex R. Army Research Institute Field Unit	Norman, Douglas O. AFRRI/DNA
Maxell, Robert L. Army INSCOM	Milbert, Allen	Novack, Roger S. BDM Corporation
Maybee, John D. DEFSMAC	Miller, Joanna A. Army Signals Warfare Lab	Novick, W. ERADCOM
Mayhew, Vicki Battelle Columbus Labs	Miller, Roger E. Lockheed Missiles & Space Co.	Obert, Luanne P. NV&EOL
Mazza, Mary E. USAMSAA	Miller, Scott F. General Electric Co.	O'Brien, Patricia N. USAMSAA
McCarthy, John Stanford University	Mintzer, Olin W. Army Engineer Topographic Lab	Ohlander, Ronald B. DARPA
McClain, Richard A. RCA Corp.	Moe, Gordon O. Pacific-Sierra Research Corp.	Olson, R. Craig MAR Associates
McDonnell, Michael M. Army Engineer Topo. Labs	Montgomery, Christine A. LOGICON, Inc.	O'Mara, Peter A. Army Soldier Support Ctr

Orlando, Nancy E. NASA Langley Research Center	Phelps, Ruth H. Army Res Inst for Behav/Soc Sci.	Remm, Robert L. ASD/DPCD (AFSC)
Orlov, Robert D. Army Concepts Anal. Agency	Piper, Admiral S. Naval Training Center	Ressler, Andrew L. MIT
Ostrum, Jean M. Advanced Technology, Inc.	Pisano, Alan D. Analytic Sciencies Corp.	Reynolds, Richard
Oswald, Robert B., Jr. ERADCOM	Pliscof, Marc S. Army Materiel Sys Analysis Activity	Richardson, Milton L. Westinghouse Electric
Overhultz, Lester M. IRAD Project on AI	Pollin, Irvin Harry Diamond Labs	Rider, James W. DEFSMAC
Pankowicz, Zbigniew L. Rome Aire Development Ctr	Poropatich, Larry T. INCO, Inc.	Robinson, Kathy I. Synectics Corporation
Pare, Roger Martin Marrietta Aerospace	Porter, Lawrence E. AFSC	Robl, Hermann Army Research Office
Pastel, Marvin P. TRADOC	Powell, Edward Gordon Naval Surface Weapons Center	Rodak, Stanley P. NV&EOL
Patay, Stephen A. TRW, Inc.	Price, William AF Office of Scientific Research	Rodman, Robert D. NCSU
Pattishall, Walter	Potka, Joseph Army Research Institute	Rogers, James L. NASA-Langley Research Ctr
Patton, J. E.	Putnam, Russell H. David Taylor NSRDC	Rogers, Leslie W., Jr. BMDATC, ATC-T
Paul, Aggrey L. Xerox-EOS	Rankin, William B. Joint Tactical Fusion Prog Mgt Ofc	Romito, Anthony Martin Marietta Aerospace
Payne, J. Roland AI&DS	Raphael, Donald L. SYSCON Corporation	Rosenblatt, Matthew A. USAMSAA
Payne, Robert W. Central Intelligence Agency	Ratches, James A. NV&EOL	Rosenfeld, Azriel Imtech, Inc.
Peperone, Salvador J. Harry Diamond Labs	Reierson, James D. MITRE Corp.	Ross, Chester C., Jr. Science Applications Inc.
Perez-Esandi, Jose J. Naval Surface Weapons Center	Reilly, James T. DOD NSA CSS	Rowell, Phillip V. JSTPS/JLDP
Pfortmiller, Lawrence G. Army Combined Arms Opns Res	Remington, Roger W. NASA-AMES Research Center	Ruberti, Robert N. Rome Air Development Ctr

Rudman, Richard	Sereno, Edgel E. AFRRI/DNA	Skevington, Richard C. RCA Labs
Rudnick, Jack J. RCA Corp.	Shaefer, Leonard A. INCO, Inc.	Slade, Stephen B. Yale AI Project
Sable, Jerome D. Advanced Technology Labs	Shapira, Joseph Bell Labs	Slagle, James R. Naval Research Lab
Salisbury, Alan B. JTF Special Task Force	Shapira, Ruth MIT, EE Department	Smith, Stephen P. Northrop Research Center
Samms, Kathy H. NASA/Langley Research Ctr	Shapiro, George Westinghouse Electric Corp.	Solo, Marcos C. NV&EOL
Sampsell, Jeffrey B. Texas Instruments	Shapiro, Linda G. Virginia Tech	Smeltzer, Stanley HQ TRADOC
Sandoris, Arthur Harry Diamond Labs	Shaughnessy, Thomas J. USACDEC	Smith, Jeff E. General Dynamics
Sarkies, Robert G. Boeing Aerospace Co.	Shen-orr, Chaim D. Government of Israel	Smith, Ora E.
Sasmor, Robert M. Army Res Inst/Behav/Soc Sci.	Shere, Kenneth D. Planning Research Corp.	Spain, David S., Jr. AI&DS
Savonarola, Dennis Ford Aerospace	Sherwood, David L. HBR-Singer	Speissbach, Andrew J. Martin Marietta Aerospace
Schaming, Bernie W. RCA Corporation	Shields, Frank J. NV&EOL	Spina, Anthony J. Rome Air Development Ctr
Schoen, Sy Litton Industries	Shields, William L. The BMD Corporation	Stayton, Lauren C. TETRA Tech Services, Inc.
Schwartz, Benjamin L. TRW Defense Systems Group	Short, Robert D. Sperry Research Center	Stainer, Howard M. DOD
Scotti, Richard S. ORI, Inc.	Sigillito, Vincent G. Johns Hopkins University	Steinheiser, Fred H., Jr. Chief of Naval Operations
Selander, J. Michael MITRE Corporation	Silva John Naval Ocean Systems Center	Stephens, Larry M. Univ. of South Carolina
Self, David W. Ford Aerospace & Comm. Corp.	Singer, G. David NV&EOL	Stevens, Albert BBN
Semon, Charles J., Jr. HQ DARCOM	Sizemore, Thomas D. Theater Nuclear Warfare Project	Stewart, Alexander Harry Diamond Lab

Stewart, Robert L.
Johns Hopkins Physics Lab

Story, David D.
Story & Kelley Smith

Stroick, Gary R.
Sperry Computer Systems

Sullivan, Julieanne K.
Westinghouse Def. & Elec. Ctr

Suttle, Jimmie R.
US Army Research Office

Swanson, Ann M.
E-Systems

Taylor, Edward C.
TRW Defense Systems Group

Taylor, Robert R.
Emerson Electric Co.

Thau, Robert J.
Litton Guidance & Control

Thibadeau, Robert H.
Carnegie-Mellon Univ.

Thole, Marsha L.
HQ DARCOM

Thomas, James J.
Battelle

Thorndyke, Perry W.
PERCEPTRONICS, Inc.

Tiede, Roland V.
Tactical Decisions, Inc.

Tindale, Larry

Tisdale, Glenn E.
Westinghouse Defense Center

Toombs, Loren E.
RCA Advances Tech. Labs

Toomey, John P.
Sperry Research Center

Tozzi, Louis
Harry Diamond Lab

Travesky, Paul D.
NV&EOL

Treadwell, George W.
Sandia National Labs

Trimble, Joel
ONR

Tseng, David
Hughes Research Lab

Tullington, Bernard J.
Battelle Columbus Labs.

Van Tuyl, Andrew H.
Naval Surface Weapons Center

Verderame, Frank
ODCSRDA

Verhey, Carl T.
Army Intelligence Ctr & School

Vieler, Eric H.
ESL, Inc.

Vitols, Visvaldis
Rockwell International

Voss, Robert A.
White Sands Missile Range

Wachter, Ralph F.
APL

Wagoner, Larry

Wallace, Gregory K.
Analytic Sciences Corporation

Walsh, Dale

Walter, Clarence P.
NV&EOL

Waltz, Edward L.
Bendix Comm Div.

Webber, Bonnie L.
Univ. of Pennsylvania

Weigle, Robert E.
US Army Research Office

Wells, Marilyn T.
Research Fellow

Whelchel, Bronel R.
CENSEI

Whinston, Andrew B.
Purdue University

Whitehead, James M.
RCA

Wickham, Connie B.
Army Engineer Topo Labs

Wilson, Susan L.
Boeing Aerospace Co.

Winkelmann, Phillip A.

Wisher, Robert
DOD

Wolff, Stephen S.
Army Ballistic Missile Lab

Wugofski, Eugene T.
Pacific-Sierra Res Corp.

Yang, Joseph H.
Westinghouse Electric Corp.

Yelowitz, Kalman L.
Ford Aerospace & Comm Corp.

Yin, Barbara H.
Ford Aerospace & Comm Corp.

Yuschik, Matthew
Univ. of South Carolina

Zarwyn, Berthold
US Army ERADCOM

Zegel, Ferdinand

Zelinski, Joan

Zemany, Paul
Sanders Associates, Inc.

Zimmerlin, Timothy A.
Synectics Corporation

Zimmerman, Bruce B.
Army Engineer Topo Labs

END

FILMED

5-84

DTIC