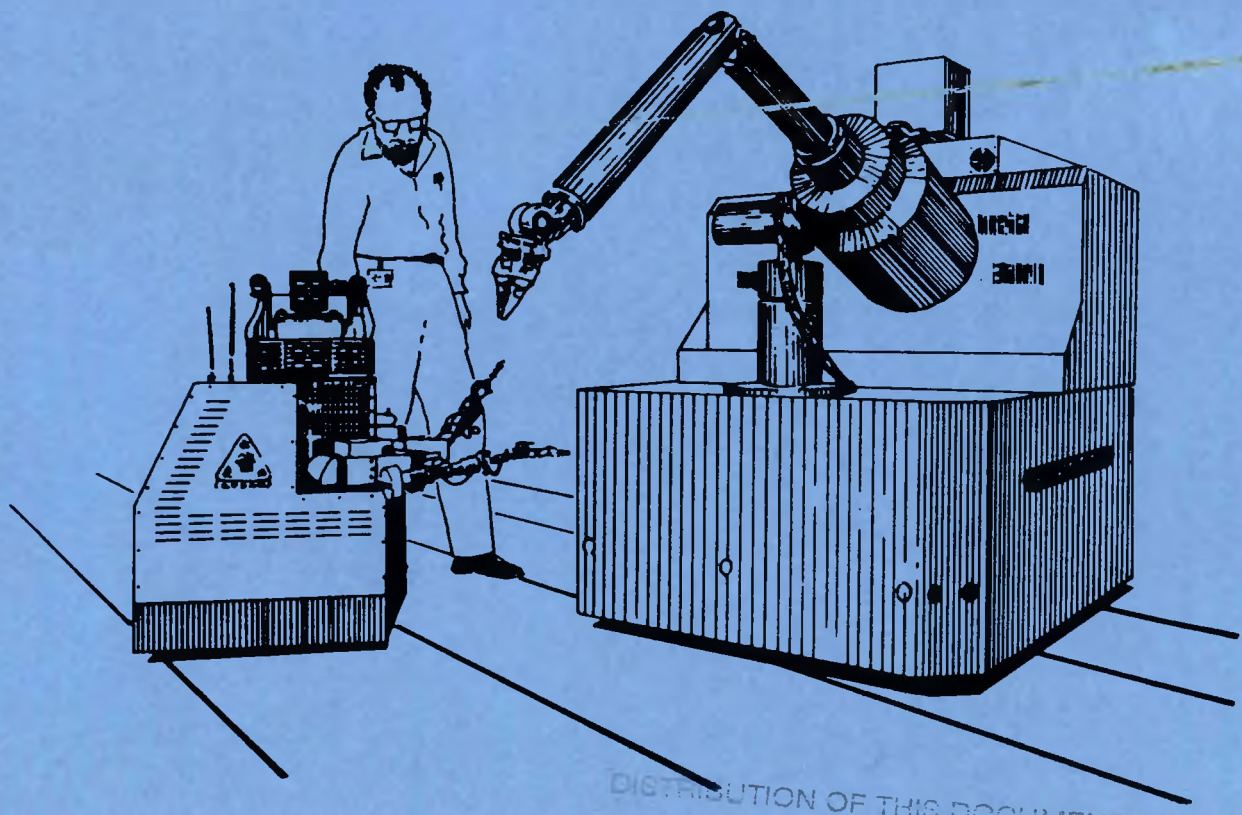
ORAU 89/C-140
CESAR-89/19

1988 Workshop on Human-Machine Symbiotic Systems Proceedings

December 5–6, 1988
Oak Ridge, Tennessee

Lynne E. Parker, Charles R. Weisbin



Oak Ridge Associated Universities is a private, not-for-profit association of 49 colleges and universities and a management and operating contractor of the U.S. Department of Energy. ORAU's mission is to foster, encourage, and engage in the identification and development of solutions to scientific, engineering, technical, medical, and human resource problems through the resources available to ORAU and its member universities. In support of this mission, ORAU provides diverse services (principally academic outreach, research, training and education, technical assistance, and technology transfer) for DOE, ORAU's member institutions, other colleges and universities, and other private and governmental organizations. Established in 1946, ORAU was one of the first university-based, science-related corporate management groups.

ORAU--89/C-140

DE89 013888

1988 Workshop on Human-Machine Symbiotic Systems Proceedings

December 5–6, 1988
Oak Ridge Associated Universities
Oak Ridge, Tennessee

Sponsored by
Oak Ridge National Laboratory
and
Oak Ridge Associated Universities

Lynne E. Parker, Charles R. Weisbin

Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences of the U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

EB

Contents

Contents (continued)

Section 1: Executive Summary

Blank Page

1988 Workshop on Human–Machine Symbiotic Systems

Abstract

This report presents the proceedings of the 1988 Workshop on Human–Machine Symbiotic Systems. Held December 5–6, 1988, in Oak Ridge, Tennessee, the workshop served as a forum for the discussion of several critical issues in human–machine symbiosis: human-machine communication, autonomous task planning and execution monitoring for heterogeneous agents, dynamic task allocation, human–machine system architecture, and machine learning via experience and human observation. The presentation of overview papers by invited keynote speakers provided a background for the breakout session discussions in these five areas. The full papers furnished by the speakers are included in the proceedings, along with written summaries of the group discussions that report session conclusions and recommendations for future work.

Executive Summary

In recent years, a growing research interest has focused on the development of methods facilitating a cooperative problem-solving relationship between humans and autonomous machines—a relationship the workshop organizers define as "human–machine symbiosis." In a symbiotic system, humans and machines cooperate in the decision making and control of tasks in a complex, dynamic environment, communicating frequently in the exchange of tasks. The function of the symbiotic system is to dynamically optimize the division of work between the human and the machine, with the ultimate goal of improving the admissible task range, accuracy, and work efficiency of the system. The successful creation of such systems requires an effective approach to several fundamental technical issues such as human–machine communication, autonomous task planning and execution monitoring for heterogeneous agents, dynamic task allocation, human–machine system architecture, and machine learning via experience and human observation.

In order to address these key issues of research and to recommend directions for future work in human–machine symbiosis, an invitational workshop was organized by Charles Weisbin and Lynne Parker of Oak Ridge National Laboratory, and by Jim Gumnick of Oak Ridge Associated Universities. The workshop was held December 5–6, 1988, at Oak Ridge Associated Universities in Oak Ridge, Tennessee. The meeting brought together 38 investigators interested in human–machine symbiosis from several research communities, including robotics, artificial intelligence, human factors, and cognitive science. To encourage informality and more open discussions, the attendance size was intentionally restricted, but participants were invited from a broad range of university, laboratory, and industry programs to provide a diversity of viewpoints. A list of the attendees and short bibliographic sketches are included in Sections 6 and 7.

To provide a background for discussion, five keynote presentations were made by internationally recognized experts: Dr. Thomas Sheridan (Massachusetts Institute of Technology) addressed the subject of human–machine communication, Dr. Avi Kak (Purdue University) presented the subject of autonomous task planning, Dr. Scott Harmon (Robot Intelligence International) spoke on the subject of dynamic task allocation and execution monitoring, Dr. Christine Mitchell (Georgia Institute of Technology) addressed human–machine system architectures, and Dr. Thomas Mitchell (Carnegie-Mellon University) presented the topic of machine learning. The full papers provided by these speakers are contained in Section 4 of the proceedings.

The keynote presentations were followed by breakout sessions whose participants were charged to develop effective research approaches and suggestions for future work in five principal technical areas. To help initiate the discussions in each of these areas, the following questions were posed to the participants in advance of the workshop:

1. Human–Machine Communication: What are the most effective means of communication between man and machine in cooperative control systems involving physical processes?

2. Autonomous Task Planning and Execution Monitoring: What are the most promising approaches to real-time task planning and execution monitoring between heterogeneous agents, at least one of which is human?

3. Dynamic Task Allocation: What are the best methods of allocating cooperative human-machine tasks?

4. Human–Machine System Architecture: What human–machine system architectures allow real-time cooperative interaction between the human and the machine?

5. Machine Learning via Experience and Human Observation: What are the most promising approaches toward providing the intelligence for a machine to learn new tasks through assimilation of experience and human observation?

Written summaries of the recommendations and conclusions of the breakout sessions were prepared by the reporters and are included in Section 5 of the proceedings. It should be noted that the summaries reflect only the opinions of the select group of workshop participants; however, the workshop organizers are hopeful that the session reports will be a valuable assistance to researchers in the human–machine symbiosis field.

It is interesting to note the attitude the attendees had concerning the definition and implications of the term "symbiosis." As defined by *Webster's Unabridged Dictionary*, and referenced by Thomas Sheridan in his paper, symbiosis is "two dissimilar organisms living together in mutual dependence." Although certain life-and-death situations (such as battlefield management) might indeed require the human to be dependent on automated machines

for survival, many of the workshop participants were uncomfortable with this implied requirement for all symbiotic systems. Instead, as presented in the Machine Learning session report by Phil Spelt, the group was more willing to accept the first definition of *Webster's II New Riverside University Dictionary 1984*: "the relationship of two or more . . . organisms in close association that may be but is not necessarily of benefit to each." The term "synergy," defined as "the action of two or more . . . organisms to achieve an effect of which each is individually incapable," was suggested as a possible alternative for describing the human–machine system. However, at least one opinion held that the symbiotic relationship could be quite beneficial by merely serving to replace humans in distasteful tasks, without requiring the symbiont to "achieve an effect of which each [symbiotic partner] is individually incapable."

Having somewhat agreed to an appropriate interpretation of symbiosis, the participants generally concurred that some degree of autonomy is a prerequisite for symbiosis, since a machine must have sufficient capabilities and intelligence to cooperate productively with a human. However, full autonomy is not required since the human can be responsible for tasks or portions of tasks that the machine is unable to perform. It was generally agreed that the tasks performed autonomously by the machine need not be performed in the same manner as would a human—that is, anthropomorphism is not required and may be an unnecessary and burdensome constraint.

An analysis of the workshop reports reveals two overriding themes recurrent in the breakout sessions: the need for the development of various types of human and machine models and the need for investigations concerning the roles the human(s) and the machine(s) should play in the symbiotic system. The development of human and machine models was recommended as an important future research topic in the Human–Machine Communication, Dynamic Task Allocation, and Autonomous Task Planning breakout sessions. The emphasis in the Communication group was for human behavioral modeling emphasizing the characteristics present during both normal and abnormal circumstances, while the Dynamic Task Allocation and Autonomous Task Planning sessions emphasized the need for the modeling of human and machine capabilities. In his paper, Scott Harmon reports on several existing techniques that have been used for internal agent modeling, while Christine Mitchell, in her paper, describes the use of an operator function model in a particular project. The participants, however, agreed that additional research is still needed in this area — both for the development of generic models and for their application to the modeling of individual operators.

The issue of the roles the human(s) and machine(s) should play in the symbiont provided some lively discussions. In most complex systems of today, a hierarchy of control exists in which certain agents (human or machine) dominate some agents while they are supervised by others. Such a hierarchy will allow humans to be controlled by non—human, or automated, components of the system. However, at the highest level of decision-making and control, current systems always place a human. The workshop opinion inclined toward continuing this system organization by allowing the human to maintain ultimate control in symbiotic systems for most, if not all, applications of the near future. The operator's associate

described in Christine Mitchell's paper reflects this superior human/subordinate machine relationship in which the computer assumes control only when the human explicitly delegates responsibility. The advantage to this approach is that by making the human the primary decision maker, he/she will have the knowledge and authority required for effective and safe system operation.

An opposing viewpoint, however, holds that the human may not always be the optimal choice for highest-level system control. The recent Vincennes tragedy in which a U.S. Navy missile cruiser shot down an Iranian airliner after mistaking it for a military fighter aircraft was mentioned as a prime example of a situation in which human interaction with a complex, time-critical, automated system is sensitive to extreme human stress and mental overload. Situations such as this may therefore benefit from allowing the automated component to assume ultimate control over the human under specified conditions. It was generally agreed, however, that at the present time no one knows how to decide when the intelligent machine should be the principal controller. Thus, these discussions led to the recognition of the need for more research on the psychological and physiological impact and the interface implications of allowing a machine to serve as the highest-level decision maker over a human.

In addition to the issues of agent modeling and the roles of humans and machines, the breakout sessions identified various other critical areas of future research and derived several interesting conclusions. The Human–Machine Communication session emphasized the need for research in the simultaneous and integrated use of multiple modalities, or sensor channels, for communication between human and machine. The goal of this research should be to make better, if not full, use of the human's sensory capabilities. Another interesting thought concerning human–machine communication held that the majority of the human's communication needs would be fulfilled with a combination of real-time visual displays and simple natural language/voice communication. The Machine Learning group determined that learning involves, in part, a transfer of knowledge from the human to the machine, and is useful when complete pre-programming is impossible. They identified two roles the human serves in symbiont learning: to act as a model from which the machine can learn (e.g. using machine vision, watching and emulating human performance), and to function as a teacher/consultant in which the human monitors the performance of the machine and makes corrections when needed. Both the Machine Learning and the Autonomous Task Planning sessions reported the need for continued research on machine learning leading to the capability to generalize. The Autonomous Task Planning group also noted the need for research to facilitate the smooth transition between human and machine control for the purpose of planning at different task levels.

The Dynamic Task Allocation session emphasized the need for measurement techniques for quantifying human and machine skills and for the determination of the appropriate task granularity that assures the smooth transition of control among elemental subtasks as they are assigned to different agents. In addition, this group noted the need for an automated monitoring facility which can dynamically assess the progress and direction of the agents—a facility also recognized as important by the Machine Learning group.

Finally, the Human–Machine Architecture breakout session determined that a symbiotic system architecture should have a human interface perspective with multiple entry points that provides user access to the various functional modules of the system as needed. One area recommended for future research involves addressing the problem of system stability under hybrid control in which the portions of the architecture controlled manually or autonomously vary dynamically over time. Another topic recommended for future research concerns the development of metrics defining quality measurements (such as effectiveness versus complexity) of a human–machine system architecture.

Judging by positive feedback both from written evaluations and verbal communication, the workshop organizers believe this 1988 Workshop on Human–Machine Symbiotic Systems was a success. The workshop served to identify key areas of research in five principal areas of human–machine symbiosis which are needed to achieve human–machine cooperative control and intelligence. The authors are hopeful that this workshop will serve as a stepping board toward advancing the state of the art in the area of human–machine symbiosis.

Acknowledgments

The authors would like to thank Oak Ridge Associated Universities staff members Jim Gumnick, Libby Kittrell, Denise Davis, Robin Stoller, Jim Pearce, and others for their contributions to the organization of the workshop. Thanks are also extended to Karen Harber for her expert typing assistance. The authors are indebted to the workshop chairs and reporters for their work in leading and summarizing the breakout sessions. Special gratitude is expressed to Dr. Joseph F. Engelberger for his prepublication review of the workshop executive summary and session report drafts. We credit Dr. Engelberger for emphasizing several points concerning symbiosis, in particular: the symbiont should not be required to achieve an effect of which either symbiont partner is incapable, anthropomorphism is not required, and the combination of real-time display and voice communication should be largely sufficient for fulfilling human–machine communication needs. Finally, we wish to thank the attendees and speakers for their enthusiastic and active participation, without which the workshop would not have been possible.

Blank Page

Section 2: Invitation

Blank Page

Oak Ridge Associated Universities
P.O. Box 117, Oak Ridge, Tennessee 37831

ornl
Oak Ridge National Laboratory
P.O. Box X, Oak Ridge, Tennessee 37831

May 25, 1988

You are cordially invited to participate in the 1988 Workshop on Man-Machine Symbiotic Systems, jointly hosted by The Center for Engineering Systems Advanced Research (CESAR) at Oak Ridge National Laboratory and by Oak Ridge Associated Universities (ORAU).

The invitational workshop, to be held December 5-6, 1988, at ORAU's Pollard Auditorium in Oak Ridge, Tennessee, will consist of keynote presentations by five internationally recognized experts: Dr. Scott Harmon will address the subject of dynamic task allocation and execution monitoring; Dr. Christine Mitchell will present the subject of man-machine system architectures; Dr. Avi Kak will address the subject of autonomous task planning; Dr. Thomas Mitchell will speak on the subject of machine learning; and Dr. Thomas Sheridan will present the subject of man-machine communication. Work sessions by the attendees will follow these talks. The primary objective of the workshop is to define effective research approaches to technical issues in man-machine symbiosis as identified in the enclosed technical scope. Answers to the following questions will be sought:

1. What are the most effective means of communication between man and machine in cooperative control systems involving physical processes?

2. What are the most promising approaches to real-time task planning and execution monitoring between heterogeneous resources, at least one of which is human?

3. What are the best methods of allocating cooperative man-machine tasks?

4. What man-machine system architectures allow real-time cooperative interaction between the human and the machine?

5. What are the most promising approaches toward providing the intelligence for a machine to learn new tasks through assimilation of experience and human

Proceedings of the workshop will be published and will include the papers presented by the keynote speakers and the reports/conclusions reached during the five breakout sessions.

In the enclosed brochure are a preliminary agenda, commitment form, and logistical information. A separate hotel reservation card and a map showing the workshop site are also included. If you agree to participate in this workshop, please return the commitment form and your $50 registration fee in the enclosed postage-paid envelope by June 30, 1988. Mark your choices for the breakout sessions on the commitment form. We will make every effort to honor your preferences, but we cannot guarantee which session you will be assigned due to the need to balance the groups.

We are intentionally restricting the attendance of this workshop to approximately 40 people to encourage more open discussions. Because of the small number of slots available and the popularity of the workshop, we must stress the importance of your attendance at the workshop if you have committed to attend. If we do not receive your response by June 30, we must assume you cannot participate, and we will invite the next person on the waiting list.

Please make your own arrangements for lodging by returning the enclosed hotel reservation form directly to the Garden Plaza Hotel before November 23. If you prefer, call the hotel directly at 615-481-2468. You will need to guarantee your accommodations if you plan to arrive later than 6 p.m.

We look forward to a successful workshop to address a timely and challenging subject. If you have any technical or administrative questions regarding the workshop, please contact Lynne Parker, workshop coordinator, at 615-574-2258.

OAK RIDGE
ASSOCIATED UNIVERSITIES

James L. Gumnick
University Relations Director

OAK RIDGE
NATIONAL LABORATORY

Charles R. Weisbin
Director, Robotics and
Intelligent Systems Program

Section 3: Agenda

Blank Page

WORKSHOP ON HUMAN-MACHINE SYSTEMS

SUNDAY, December 4

7:30 p.m. **Welcoming Reception** — Garden Plaza Hotel

MONDAY, December 5

7:30 a.m. Coffee — ORAU Energy Conference Room

8:00 Registration

8:30 **Welcoming Remarks and Introductions**
Dr. James L. Gumnick, Oak Ridge Associate Universities
Dr. Charles R. Weisbin, Oak Ridge National Laboratory

8:45 **Keynote Speech 1 — Man-Machine Communication**
Dr. Thomas B. Sheridan, Massachusetts Institute of Technology

9:30 **Questions/Answers on Man-Machine Communication**

9:45 **Break**

10:00 **Keynote Speech 2 — Autonomous Task Planning**
Dr. Avi Kak, Purdue University

10:45 **Questions/Answers on Autonomous Task Planning**

11:00 **Keynote Speech 3 — Dynamic Task Allocation and Execution Monitoring**
Dr. Scott Harmon, robot Intelligence International

11:45 **Questions/Answers on Dynamic Task Allocation and Execution Monitoring**

12:00 noon Catered Lunch - Energy Conference Room

1:00 **Breakout Sessions**

- Man-Machine Communication (Energy Conference Room)
  Bill Knee, Chair
  Marty Beckerman, Co-Chair/Reporter

- Dynamic Task Allocation (Pollard Auditorium)
  Lynne Parker, Chair
  Wayne Manges, Co-Chair/Reporter

4:15 **Break**

4:30 Reconvene in Energy Conference Room for Reports on Breakout Sessions

5:30 Adjourn

6:00 Poolside Reception — Garden Plaza Hotel

TUESDAY, December 6

8:00 a.m. Coffee — ORAU Energy Conference Room

8:30 **Introduction**
Dr. Charles R. Weisbin, Oak Ridge National Laboratory

8:45 **Keynote Speech 4 — Man-Machine System Architecture**
Dr. Christine Mitchell, Georgia Institute of Technology

9:30 **Questions/Answers on Man-Machine System Architecture**

9:45 **Break**

10:00 **Keynote Speech 5 — Machine Learning via Experience and Human Observation**
Dr. Thomas Mitchell, Carnegie Mellon University

10:45 **Questions/Answers on Machine Learning via Experience and Human Observation**

11:00 **Breakout Sessions** — Pollard Auditorium

- Man-Machine System Architecture
  Bill Hamel, Chair

- Machine Learning via Experience and Human Observation
  Francois Pin, Chair
  Phil Spelt, Co-Chair Reporter

1:00 p.m. **Working Lunch** — Energy Conference Room

2:00 Reports on Breakout Sessions

3:00 Wrap-up

4:00 **Adjourn**

###

Blank Page

Section 4: Keynote Presentations

Blank Page

Man-Machine Communication for Symbiotic Control

Thomas B. Sheridan

Man–Machine Systems Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02138

Abstract

As telerobotic and other systems are asked to do more sophisticated tasks, the communication between man and machine becomes more critical. This paper specifies communication functions and forms both of human and computer, discusses telerobotic control activities about which communication is essential in relation to the human's mental model and the computer's internal model, gives examples of critical research areas in man–machine communication, and suggests appropriate performance measures.

Delimiting the Problem of Man–Machine Symbiotic Control

J.C.R. Licklider, in his important "Man-Computer Symbiosis" [1] used as his example of the term *symbiosis* the larvae of *Blastophaga grossorum*, which lives in the ovary of the fig tree and is itself responsible for pollinating the fig tree. By analogy, Licklider foretold the needs and problems we would have in trying to engineer symbiotic man–machine systems. He describes the problems of having computers "participate in formulation and real time thinking," the need for "time and motion analysis of technical thinking," and the difficulties of developing computer languages "not for specifying course" but for teaching the computer "goals, as in human communication." He mused on "desk surface display and control" (this was long before the mouse emerged!), on "computer-posted wall displays" and on automatic speech generation and recognition—all avenues of communication that appear salient for us today.

One might ask whether man–machine *symbiosis* (*Webster's Unabridged*: "two dissimilar organisms living together in mutual dependence") is more apt today than earlier? Years ago, at least in some contexts, we might have characterized man as totally dependent upon his tools, but we wouldn't think to characterize the tools as dependent upon man for survival. Today, however, the computer, or now more generally the *intelligent and at least semi-autonomous* robotic system, may be considered metaphorically to have a life apart from its operator, and in that sense we may use the idea of a symbiotic relation. We think of communication in a such a symbiotic relation as necessarily being easy and reliable.

Historically, formal experimentation in modern human–machine communication may be said to have begun with telephony. As the telephone matured, the Bell Laboratories became a font of research in human–machine communication, not only machine-mediated human–human communication but serious research on dialing and punching buttons. At

about the same time, World War II called attention to human–machine communication in aircraft cockpits and ship command stations. Along the way engineers who design chemical plants and other process and manufacturing automation became concerned about man–machine communication in control rooms. Then the stakes increased dramatically as digital computers became available to everyone, and microcomputers gradually found their way into all kinds of businesses and factories. The computer has become an integral part of man–machine cooperation in almost all applications, ranging from automobiles to air traffic control, from banks to hospitals, from home appliances to large scale chemical and nuclear plants. So man–machine symbiosis, insofar as it does or can exist, is really human–computer symbiosis.

The latest concern is human–computer symbiosis (and consequently communication) for teleoperators and telerobotics—which, at least in part, is the subject of this meeting. By *teleoperator* I mean a machine which has sensors and actuators, performs useful work on its environment, and is controlled remotely by a human operator. *Direct teleoperation* means the human continuously controls every action, much as a puppeteer controls a puppet. A *telerobot* is a teleoperator which also embodies understanding, memory, and decision capability so that the human operator, as a *supervisor*, may communicate to it high-level goals and contingencies and receive high-level state information, while the machine executes low level functions and pieces of the task semiautonomously by closing the loop through its own effectors, sensors and internal computer.[2] Figure 1 illustrates supervisory control of a telerobot. A single human operator may supervise multiple telerobots of sufficient autonomy. The reason the human operator has an important role to play is that the robot cannot generally be trusted in uncertain environments; human operators are needed to plan for the telerobot, teach it, monitor its autonomous activity, take over in case of failure and reprogram it as necessary, and learn from the experience.

Functions and Forms of Man–Machine Communication

Communication at the human–machine (human–computer) interface has two principal functions (Figure 1): communication of the human operator's *intent* in the form of *commands* to the machine (diagrammed by the four steps in the upper block of Figure 2) and communication of the machine's *state* (or the task situation) to the human (diagrammed by the four steps in the lower block of the figure). For each function there is communication in two directions, where the secondary direction is for clarification and feedback about that function, thereby constituting a closed loop, or a dialog. Each of these four communications (1 to 2, 3 to 4 in each of the two blocks) involves active participation by both human and computer in an appropriate sender or a receiver role, as indicated by the phrases at the left or right of Figure 2. Further, each of the four communications can take either *analogic* and *symbolic* form, or a combination. None of these functions, directions, or forms is independent of the others; indeed they are closely coupled.

Communicating intent: flexible interactive command language and feedback of understanding

Probably the most primitive form of command language is analogic, wherein the hand or other parts of the body move relative to or apply forces on the environment. Indeed these motions or forces may be performing a task directly. Normally we might not think of this as communication, but nature understands it as communication and necessarily complies according to her rules, the laws of physics. We may engineer the environment (machine) so that the motions/forces are operating valves, switches, knobs, steering wheel, etc., to activate the machine in some appropriate way. In that case it is easier to think of the motions/forces as *signals* for communication of information, though still carrying necessary energy to operate the machine. Human factors studies as well as common sense have shown the importance of arranging things so that the motions/forces correspond in some geometric or physically isomorphic way, both to the form of sensory feedback available (they call it *stimulus-response compatibility*) and to the way we normally think about what the signal means (*the population stereotype*). Turning the steering wheel so that its top moves in the direction we want the car to go and moving the master arm or joystick in the direction we want the slave arm to go are examples of the former. Throwing the switch up for "on," down for "off," and rotating the knob to the right for increasing the variable are examples of the latter. Sometimes the stereotype is clear, but sometimes it is not. For example, most of the world operate their light switches in the opposite direction. All of this points up the importance of the so-called *mental model*, which we shall come back to.

Symbolic command language consists of spoken words or sounds or typed alphanumeric characters concatenated according to agreed upon rules. Current linguistic theory hypothesizes that the sophisticated computational architecture required of our brains to produce such natural language is built into our genes and not simply learned.

When communicating with other people, whether through body language or spoken/written language we make many assumptions—both about the specific domain of the task at hand and about "universals" such as relevance, consistency, cooperation, etc., which essentially form the rich community or cultural context that sets a norm for all of our behavior. Such norms not only establish the rules governing how symbols are concatenated by the sensor so as to be understood by the receiver (syntax), but also determine what terms to apply to what different objects or events to convey meaning (semantics) and how to relate to existing circumstances with regard to actions to be taken (pragmatics).

A computer which is not programmed with all of these constraints cannot possibly decide what the received message intends for it to do. This is not a serious problem when by prearrangement each simple discrete message maps to a unique action independently of all other messages and actions, but it poses a very difficult problem when the sender must give orders by concatenating discrete commands from a relatively small message set in order to have a robot perform acceptably in a partially unknown environment with a multiple infinity of alternative actions. Thus, curiously, more constraints enable richer communication and, therefore, more freedom of communication than do fewer constraints!

Human to human communication of intent is mediated by the receiver's feedback in some form that the message is being understood or not—nods of the head, interjected "uh-huhs" etc. In a telerobot system, display of this information can be graphical or alphanumeric. It must convey to the sender (the human operator) a "picture of the computer's model" for what is to be done. Artificial intelligence (AI) and computer science are hard at work on aspects of the understanding and feedback problem. One salient approach, called *plan recognition*, is where a goal (explicit or implied) and a starting point are communicated to a computer and the computer pieces together from a data base of "if . . . then" statements (*production rules*) a specific plan for reaching the goal.[3] Another approach, called *plan generalization*[4] is where a goal, a starting point, and some steps along the way are given and the computer generalizes from this what to do in a slightly different situation later (to save the operator from having to repeat all the details again or in case the initial plan execution encounters trouble and the computer needs to step in and help). A third approach, called *state space reduction*[5] is where the human operator gives a few steps along the way for the purpose of delimiting the huge hyperspace of alternative actions sufficiently that the computer has enough memory to search and optimize the final trajectory according to prearranged objectives.

Currently available command languages for industrial robots such as VAL[6] are close to the unique command-action mapping mentioned above but have limited capability to allow short concatenations of symbols to work as more general subroutines or to be redefined as *chunks* or *macros*. We are still a long way from instructing a computer much as we would instruct another person,where the computer can indicate that it understands one part but is confused about another, and the human teacher can respond that the computer seems to be missing a particular point and provide an example, or express the command in another way, i.e., where there is a dialog about what the human operator wants the computer to do relative to what the computer can understand and can do.

Communication of system state: multimodal adaptable display

Information about system state may also be analogic or symbolic or both. The conventional state displays for telerobots are video, computer generated alphanumerics, and perhaps some dedicated warning lights and a dial or two; however, much more will be demanded in the future.

First, vision is not the only sensory mode of interest. Consider audition. Computer-generated speech is here today but unfortunately is not being exploited much. It is being used for telephone systems, automobiles and public transit, and point-of-sale machines in rather trivial ways. There is need to explore its full usefulness and what its limits are. We also know a lot about binaural sound localization which might be used for generating three-dimensional analogs of spatial configurations.

There also is new understanding and interest in computer-generated displays of force to the human hand-arm muscles which correspond to gross or resultant forces applied by the task to the robot hands. There is corresponding need to provide tactile dispays—presentations of

patterns of force and/or displacement to the human operator's skin, based on corresponding patterns applied to the robot hand. Tactile display might also be accomplished visually or presented to a part of the body other than where the master arm or joystick is being held.

Vision remains the sense most amenable to communicating large quantities of interrelated data, especially about spatial relations. Computer displays are continuing to become higher resolution. Stereoscopic and three-dimensional displays are being experimented with. Memories are getting bigger and faster so that superposition of computer-generated graphics on video is now relatively easy. What we do not know is how best to employ all these newfound capabilities.

Both query by the operator about system state and clarification by the computer might best be accommodated by using the computer-graphic display as a common "blackboard," where both "point" to physical objects, steps in a plan, or other parts of pictures or diagrams and either voice or an abbreviated alphanumeric dialog language is used to arrive at mutual understanding.

What Man and Machine Need to Communicate About in Order to Do Symbiotic Control

Having laid out the requirements of communication (of human intent to the computer and system state to the human), it is now appropriate to consider the things that human and computer must communicate about in the context of jointly performing telerobotic tasks. The reason for doing so, as implied above, is that the operator's mental model (the pattern or configuration of causation or correlation, remembered as words or images or in other ways, which the human operator claims he associates with given objects or events) and the computer's internalized model for the similar interrelations among those same variables must be consistent with each other. Insofar as possible, the command language and display language should then embody a corresponding form. It goes without saying that all four internal models should correspond to the task in space, time, and other key variables. Figure 3 illustrates the idea.

What, then, are the categories of communication activities for a telerobot system? I propose to divide them into two classes: *off-line* and *on-line* activities (see Table 1). Off-line activities are: (1) learning the physical properties of the telerobot and task and learning whatever lessons there are from experience in past operations, (2) exploring and setting trade-offs among conflicting incommensurable objectives by *satisficing*, i.e., finding feasible alternatives which are "good enough" according to subjective criteria which the operator cannot easily explicate, and (3) strategic planning, i.e., the development of general and contingency plans and guidelines. On-line activities are: (4) allocating sensors, i.e., specifying what information gathering will be done, given that because of time, risk, and cost constraints sensors can attend to but a small subset of what is available to look at, (5) monitoring for failure or abnormality by use of specialized filters tuned to particular symptoms as well as general *topographic* methods to detect any deviation from normality, (6) making control decisions, using standard techniques from modern control theory perhaps

as well as newer techniques of fuzzy control, and (7) implementing control decisions, i.e., telling the computer what is to be done in concise but unambiguous language and making sure it understands.

For each of 1–7 in Table 1, the human operator has appropriate mental models, and for each the computer also must have some internal model and stored data. For each, the human must advise the computer of intent, and the computer must advise the human of what it infers from its internal model—what it "knows" about the ongoing activity and its relevance to the system. In addition, each can query the other and give advice in response to the other's query (or give advice solicited by a priori query such as warning or alarm information). Thus in each of the seven categories the human and a computer-based "expert system" may be said to be in dialog. Table 1 provides a phrase or two describing the appropriate computer internal model and the corresponding human mental model in each case. Naturally the computer should be relegated to the algorithmic functions such as equation solving, storing, and making available quantitative data. The human operator, by contrast, is best at intuition and associative memory and recall.

Examples of Critical Research Areas in Man-Machine Communication

Ordinarily some of the following are not considered communication problems. Actually they are. Human–computer communication is at the core of each.

Exploration of multiobjective Pareto alternatives

Perhaps the ultimate reason to retain human operators in complex systems is to decide what trade-off of objectives is most appropriate or at least what small subset of alternatives is acceptable *(satisficed)* in each task; eventually everything else may be automated. Whether we ever reach such a state of automation is doubtful, but there is an increasing awareness of the operator's need to compromise among multiple incommensurable objectives and greater interest in providing decision aids for this purpose. One approach to such decision aiding is to abandon the notion that there exists or that the operator can produce anything resembling a global utility function, other than to specify weak order with respect to each objective. Assuming the operator or some other data source(s) can input to the computer the set of feasible alternatives, the computer can then specify the dominant subset—the Pareto frontier—those that are better in one or more objectives and no worse in others (illustrated in Figure 4). The problem then is largely one of flexible communication, where the operator explores the implications of adjusting weights and "least acceptable" constraints in each of the objectives. Laboratory experiments[7] have shown that the analogic or symbolic communication form by which the operator makes the adjustments and sees the implications is critical.

Adjustment of impedance

The common *position control* mode of teleoperation means positioning a multidegree-of-freedom "master" arm, having the "slave" arm servoed so as to adapt a corresponding position in 6 degrees of freedom and getting feedback to the hand and arm muscles of whatever force

is required to maintain that *position* correspondence. By simply adjusting control gains, such a system can instead be made to impose the same *force* as applied to the master, letting the position take whatever values it must. In fact, as Raju[8] and others have shown, the force-position relationship (*compliance*, or when damping and inertia area also factored in, *impedance*) may be adjusted to be different in each DOF and/or at each of master and slave ends, depending on the task requirements and the sensitivity and strength of the operator. However, if gains are not adjusted carefully, oscillations will result, seeming to impose noise on the analogic communication.

Touch

Touch is obviously an important means of communication with the environment for animals and man, but we have hardly begun to understand how to use it in human–machine systems. We now have crude touch sensors for robots. Experiments have been conducted on primitive displays to the skin, but when that same skin is communicating in the forward loop (applying forces or displacements or, which is the same, engaged in a "resultant force" impedance relation with a telemanipulated object) other tactile information patterns are masked. For this reason researchers have sought to display contact or force patterns from tactile sensors to skin of other parts of the body, or convert these patterns to light and display them to the eyes[9] or generate corresponding sound patterns for the ears, etc. The field is wide open at this time.

Telepresence

Presumably the term *telepresence* means "a feeling of being present at a location actually distant." There now is faith that telepresence will help teleoperation. Indeed experiments with head-mounted video systems (which is probably the most striking way to generate that feeling) would seem to bear this out. Gross force feedback to the muscles and touch feedback to the skin are also regarded as essential to high-fidelity telepresence. Curiously, no one has ever shown that the sense of presence *per se* contributes a thing, apart from providing the control feedback needed to perform the task (which may be provided without telepresence). With progress in higher resolution, reduction in size, and lower costs for video and other sensors, we can expect some of these questions to be put to critical test.

State estimation

State estimation means combining (1) the latest (usually noisy and delayed) measurement data with (2) the predictions of an internal model about what *should have* happened, given the last estimate of what *was* happening earlier and knowledge of what control signals have been input to the system, to make a "best" (if all the assumptions are true) decision about what *is* happening *now*. Computers are good at state estimation for noisy data and reasonably good internal models of what variables are relevant and how they interact. People are poor at state estimation for purely deterministic situations like tumbling satellites but much better than computers at second-guessing the relevance of variables which are hard to model. Getting human and computer to communicate and cooperate in this task poses certain difficulties,[10] like the fact that the computer can spit out large multi-dimensional probability densities which the human cannot comprehend so that much information must be discarded first. The state estimation (or, more generally, situation assessment) problem is

seen as a challenging one for human–computer symbiosis.

Teaching relative commands

One way to teach a robot is to demonstrate a procedure (analogically) or to specify its course point-by-point numerically (symbolically) and then have the robot perform. A second way is to specify a set of points at key way-points, much as air traffic controllers communicate to pilots, and then the pilot is responsible for employing criteria agreed to beforehand to get from point to point. A third way is to specify the constraining geometry, the goal , and salient criteria in addition to those given earlier, and then say, in effect, "OK, do it." Each method expects a bit more intelligence of the computer, but, as Yoerger[11] has shown, it need not take much and one can achieve both speed and accuracy by letting the computer do what it does best. Communicating human delays and noise accomplishes nothing.

Computer understanding

Computer understanding is a very big undertaking, as suggested in the discussion above; nevertheless, in consideration of ensuring that human command language works, it may well be the emphasis which has been neglected for too long. It is now an active field in AI, and practical applications are beginning to emerge.

Multi-DOF, redundancy

A six degree-of-freedom arm can attain any commanded end position and orientation within range, but the position and orientation of all intermediate links, are then fixed. But with a few additional links the arm can reach the desired end position and orientation and also avoid given obstacles. Das[12] has developed a technique for accomplishing this, where the human operator tells the computer where the obstacles are and guides the arm's end point through the obstacle field while the computer forces intermediate joints to avoid the obstacles and alerts the operator if it cannot. Part of Das's system is a technique for automatically selecting a "best" viewpoint for the operator.

Span of autonomy; communication policy

By *span of autonomy* is meant, loosely, the interval of time, space and/or instructions during which the telerobot is on its own. Actually, there are several "spans," and their interrelationship must be established as part of a *communication* policy. Table 2 lists six types of message for both human-to-machine and machine-to-human communication. From top to bottom they progress more or less from "normal" to "emergency." One would expect the messages at the top of each list to occur more often than those at the bottom. Those toward the bottom are less expected and higher priority, and deserve to take precedence over those toward the top. Which of computer or human has ultimate control *when* is a politically tricky issue. Communication policies need to be worked out for any particular system and mission, but it is clear that there is no single span of autonomy; there are multiple spans depending upon circumstances.

Deception by computer

Human–computer communication can be deceptive. Weizenbaum[13] makes clear the ways in which people can be fooled into thinking computers are communicating sense when actually they are communicating nonsense. An example is Weizenbaum's own *Eliza* program, which convincingly simulates a psychiatrist (for several minutes before one begins to suspect that the computer isn't very smart) by picking up words and phrases and repeating them in plausible syntax. In the writer's opinion more anticipatory research of this kind needs to be done.

Performance Measures of Man–Machine Communication

Any improvement in human–machine communication must ultimately prove itself in actual use. It is reasonable, however, to require demonstration of improvement long before actual operation of any given system, namely at the stage of applied research and early system development. At this stage it is desirable to have some measures which can be applied to show the extent to which any one approach is better or worse than any other.

Information without meaning

One such measure is classical information theory.[14] During the 1960s, this was applied extensively by psychologists who were caught up in it as a new fashion, but just as abruptly it was fell from fashion, for indeed it was no silver bullet. Given any matrix with categories of what the human operator intends to communicate on one axis and what is understood on the other axis, and assuming communication is not perfect, there is a *confusion matrix*, which is formally amenable to information transmission analysis. Such measure, however, cares only about reducing consistency—the receiver's uncertainty about which message was sent—quite apart from any meaning any message happens to have.

Value of information

If, when information about circumstance x is communicated perfectly and on time, one could take that action u which nets the greatest gain for that x, the expected gain (G) would be

$$E_1(G) = \int_x \max u\ [G(u,x)]\ p(x)\ dx$$

where G(u,x) is the value of action u contingent upon circumstance x. If one had to select an action without knowing x except by its prior probability density p(x), then one could do no better than

$$E_2(G) = \max u\ [\int_x p(x)\ G(u,x)\ dx]$$

The *value of the information* about x is then

$$E_1 - E_2$$

Relative membership with respect to a fuzzy rule

A third measure of communication, and one explicitly tied to linguistic meaning, comes from fuzzy set theory.[15] A fuzzy set, unlike a crisp set to which an object or event is a full member or not a member, has degrees of membership from 0 to 1, depending on the value of an argument (x). Figure 5 illustrates several fuzzy variables ("short," "medium," etc.) for the physical argument "height" and several for the argument "girth." (It is assumed that a person can generate such membership functions.) In terms of these variables, x, which has physical values of height and girth, may also be expressed as a fuzzy vector (0.4 very tall, 0.9 tall, 0.6 medium, 0.0 short, and similarly for the fuzzy variables skinny, normal, and fat). For any given rule stated with Boolean "and" and "or", one can substitute "min" and "max", respectively, and infer the relative membership of every x. One may then select the x with the greatest membership as the best choice (for president or whatever). The point is that membership is inherently a measure of relevance to a stated rule or set of rules, where the rule(s) can be expressed in everyday words. When a given x fits perfectly, we may say it is a very good (true, relevant) instance of the stated rule (expressed in fuzzy words). We have, therefore, a goodness-of-fit metric by which to tie the meaning of words to physical objects or events. It is not unthinkable that the "terms" of body language and other forms of communication can provide such fuzzy variables relative to physical arguments and that the meaning of concatenations of body language can be evaluated in the same manner.

User friendliness

User friendliness is a term often employed in characterizing human–computer communication (user friendliness presumably produces good communication). There are no accepted definitions as yet. Perhaps fuzzy measures are appropriate or maybe just old fashioned rating scales.

Conclusions

Various considerations regarding communications between man and machine (more specifically human operators and computers in telerobot control tasks) have been presented, toward the end of more cooperation (symbiosis) between man and machine. Communication functions and forms of both human and computer are specified. Activities within teleoperation and telerobotics, about which communication is essential, are discussed, particularly in relation to the human's mental model and the computer's internal model. Examples of critical research areas in man-machine communication are given, and different types of performance measures for man-machine communication are described.

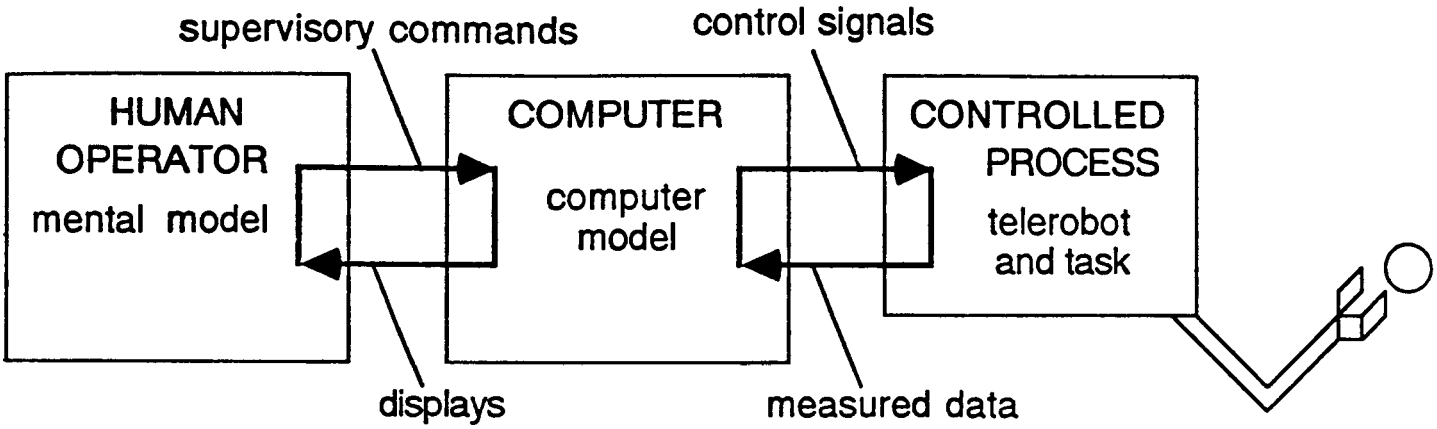


Figure 1. Supervisory Control of a Telerobot

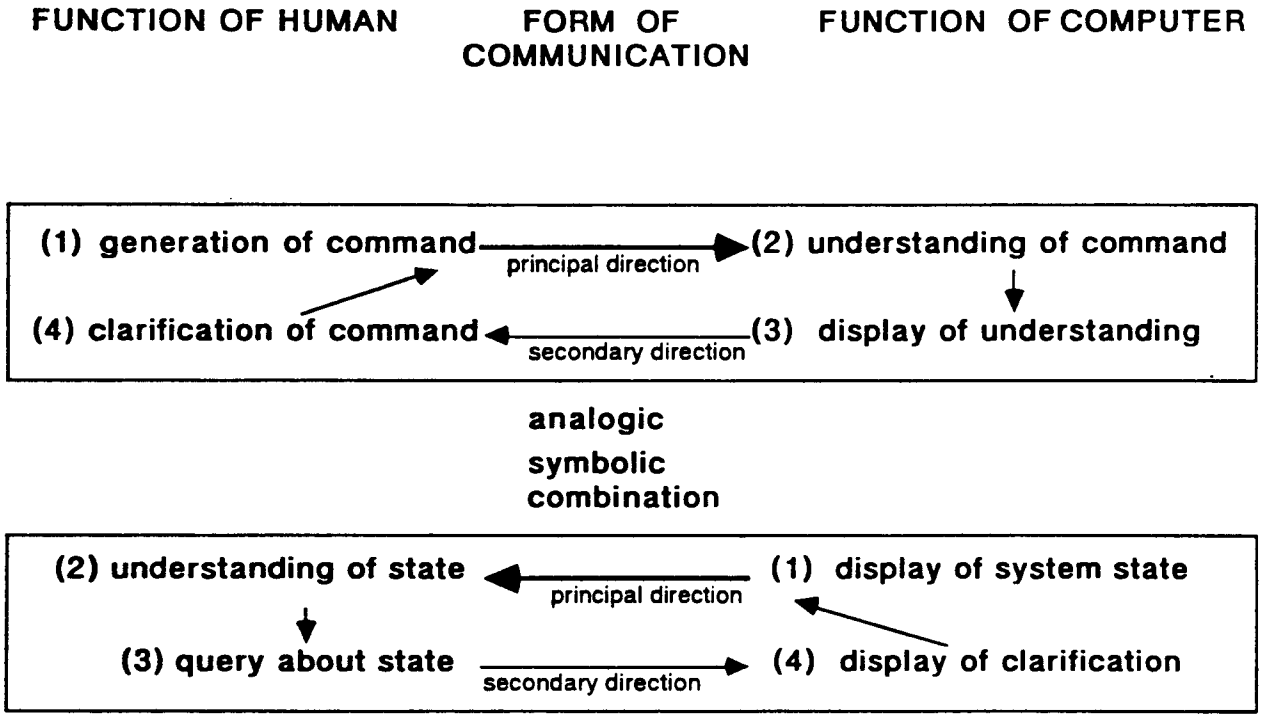


Figure 2. Functions and Forms in Human—Computer Communication

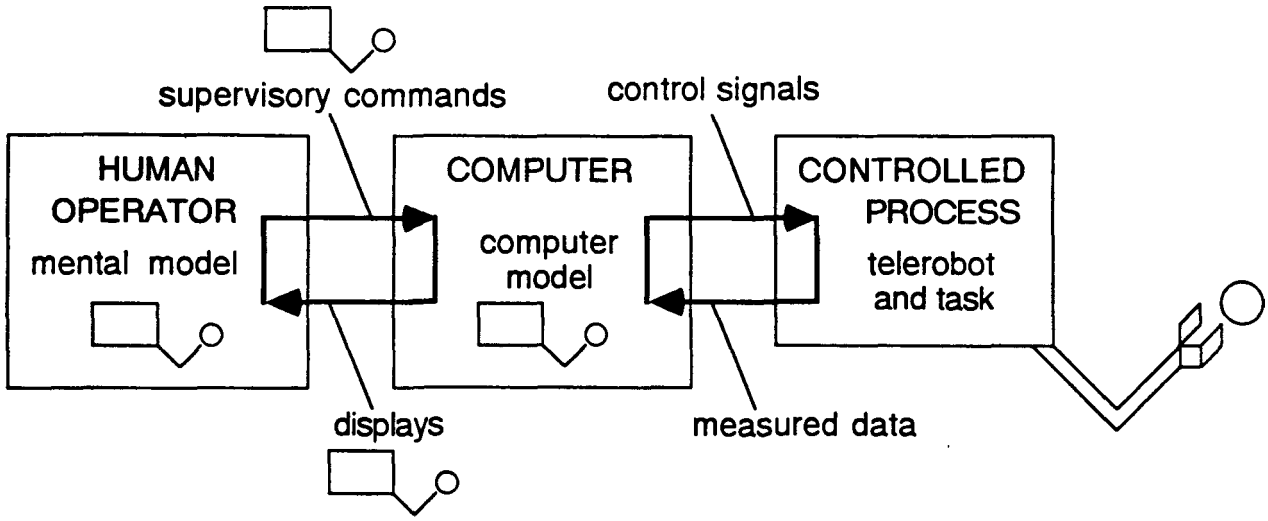


Figure 3. Models of Controlled Process Internal to Human, Computer, Command Language and Display

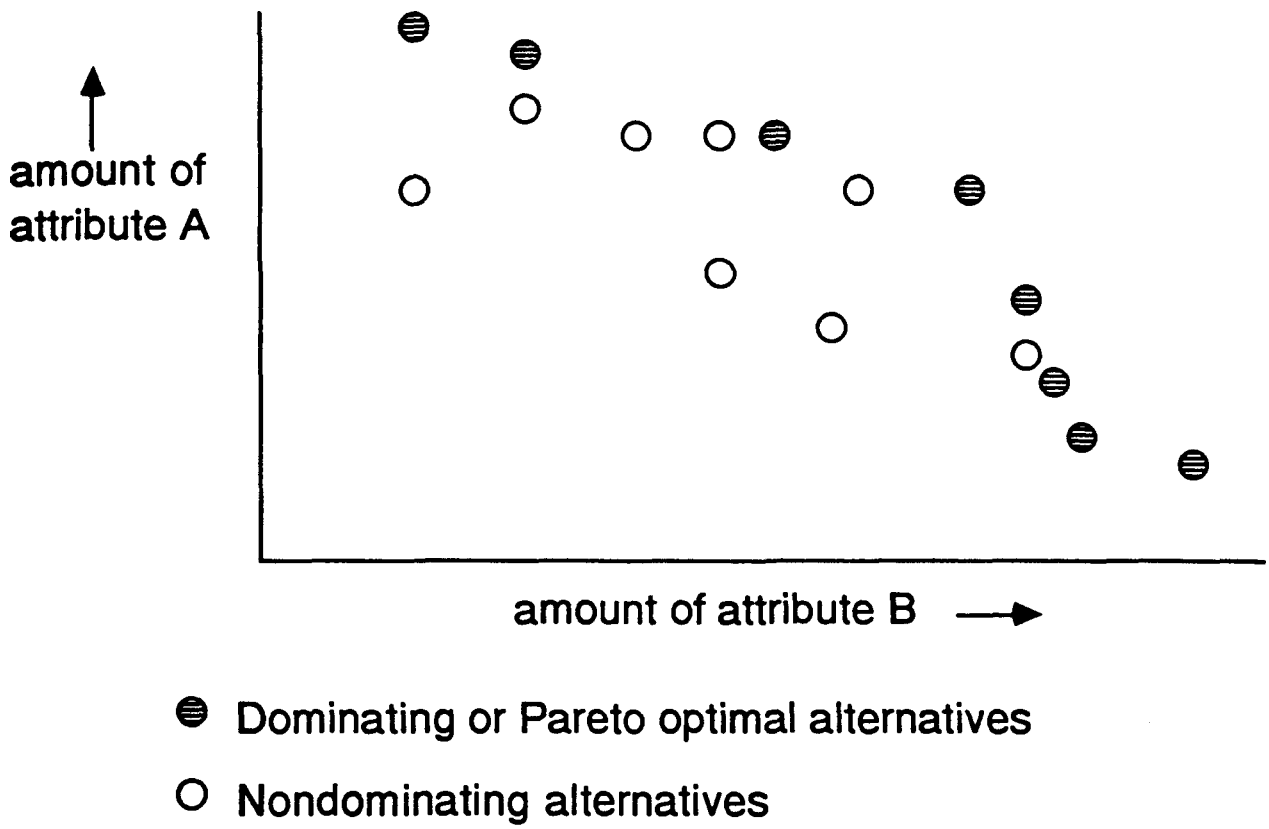
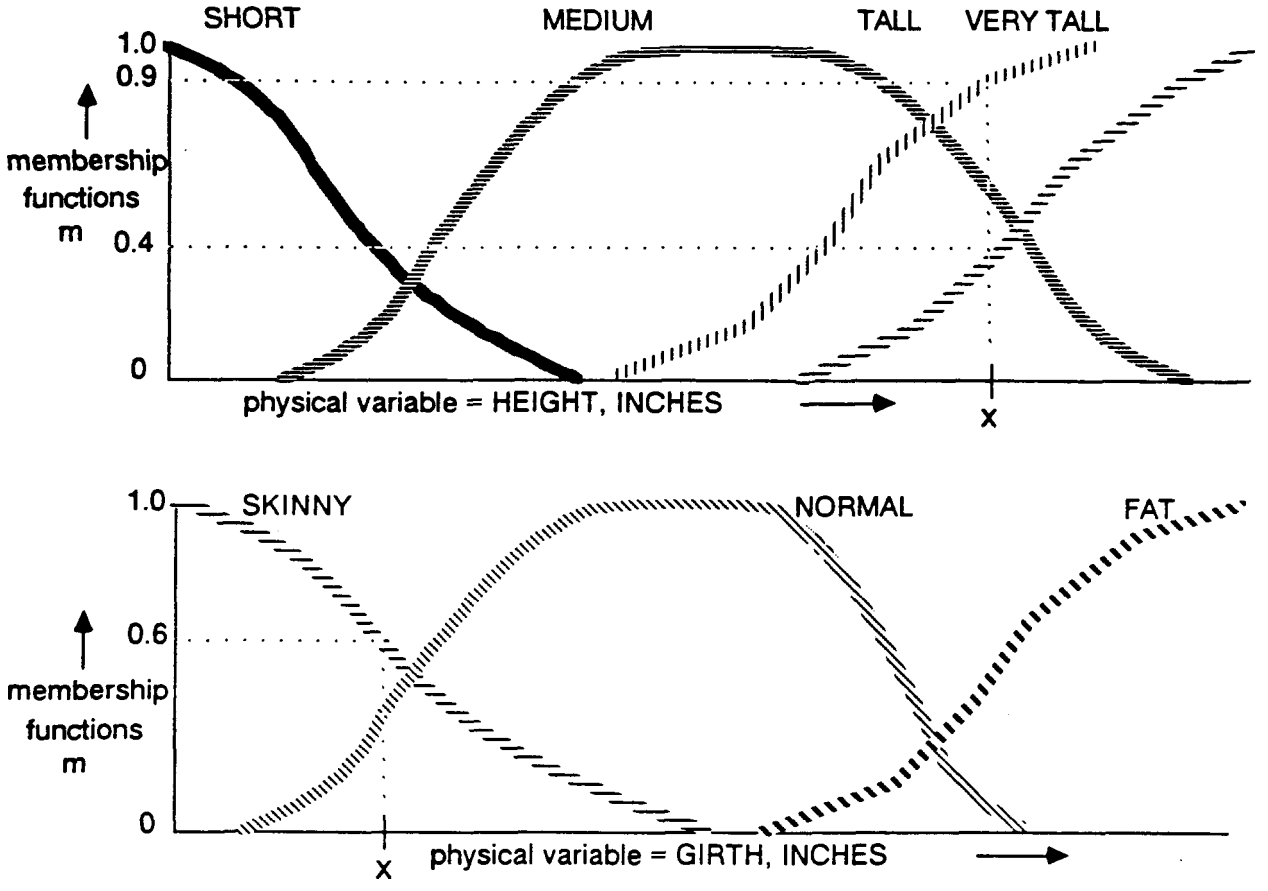


Figure 4. The Pareto Optimal Frontier of Alternatives to be Explored and Satisficed



Rule:
If candidate is tall or very tall, and is skinny, or if s(he) is short and fat, then reject.

Parse:
If { [(tall) or (very tall)] and [skinny] } or { [short] and [fat] } then reject

Evaluate membership for each person, e.g. for x
(using or = max, and = min)

max [(tall), (very tall)] = max [0.9, 0.4] = 0.9,
min { [tall or very tall =0.9] , [skinny = 0.6] } = 0.6
min { [short = 0], [fat = 0] } = 0
max ({tall or very tall, and skinny = 0.6}, {short and fat = 0}) = 0.6 for rejection

If have other rules, apply them also to every x,
take action with greatest membership

Figure 5. An Example of Fuzzy Variables and How Membership is Established in Relation to Rules Stated in Terms of these Variables.

ACTIVITY REQUIRING COMMUNICATION	COMPUTER INTERNAL MODEL OF (in relation to system state)	HUMAN MENTAL MODEL OF (in relation to intent)
off-line activities		
(1) learning	physical equations of telerobot task, record of past experience	intuitive physics of telerobot task, lessons from past experience
(2) exploring and setting objectives	tradeoffs among constraints, Pareto alternatives	acceptable constraints, choice from Pareto alternatives
(3) strategic planning	procedures compatible with (1) and (2) above	laws, institutional policies, cultural norms that apply
on-line activities		
(4) allocating sensors	availability, constraints, calibration of sensors	expected return, trustworthiness of sensors
(5) monitoring for abnomality / failure	symptoms of abnomality, causal topography	symptoms of abnormality, deviations from normality
(6) making control decisions	observer, state estimator, control law	satisfaction of objectives, intuition and skill in control
(7) implementing control decisions	syntactic rules for specifying command language	semantic, pragmatic associations, efficiency of communication

Table 1. Computer and Human Internal Models Associated with Various Telerobot Communication Activities

HUMAN TO MACHINE	MACHINE TO HUMAN
normal preoperational teach	normal operational feedback
update program according to plan	request for program update according to plan
respond to request for clarification	request for clarification
interrupt based on reconsidered or updated endogenous information	interrupt based on local computer decision
interrupt based on exogenous input	interrupt based on uncontrollable or unknown cause

Table 2. Messages which are Treated Differently According to Communication Policy

References

[1] Licklider, J.C.R. (March 1960), *Man-Computer Symbiosis*, Vol. 1, No. 1, paper 1 of IRE Transactions on Human Factors in Electronics March 1960.

[2] Sheridan, T.B. (1988). Telerobotics, to be published in *Automatica*.

[3] Allen, J.F. and Perrault, C.R. (1980), Analyzing intention inutterances, *Artificial Intelligence*, Vol. 15, 143–178.

[4] Yared, W. (1988), Cambridge, MA: MIT Man-Machine Systems Lab., PhD Thesis, in progress.

[5] Park, J. (1988), Cambridge, MA: MIT Man-Machine Systems Lab., PhD Thesis, in progress.

[6] Lozano-Perez, T. (1983), Robot programming, in *Proc. IEEE*, Vol. 57, No. 7, 821-841, July.

[7] Charny, L. (1988), Cambridge, MA: MIT Man-Machine Systems Lab., PhD Thesis, in progress.

[8] Raju, G.J. (1988), *Operator Adjustable Impedance in Bilateral Remote Manipulation*, Cambridge, MA: MIT Man–Machine Systems Lab., PhD Thesis.

[9] Bejczy, A. (1983). Sensors, controls and man-machine interfaces, *Science*.

[10] Roseborough, J.B. (1988), *Aiding Human Operators with State Estimeation*, Cambridge, MA:MIT Man–Machine Systems Lab., PhD Thesis.

[11] Yoerger (1982)

[12] Das, H. (1988), Cambridge, MA: MIT Man-Machine Systems Lab., PhD Thesis, in progress.

[13] Weizenbaum, J. (1976), *Computer Power and Human Reason*, San Francisco, Freeman.

[14] Shannon, C.E. and Weaver, W. (1963). *Mathematical Theory of Communication*, Urbana, Univ. of Illinois Press.

[15] Zadeh, L. (1984), Making computers think like people, *IEEE Spectrum*, Aug.

Blank Page

A Task Planner for Simultaneous Fulfillment of Operational, Geometric and Uncertainty-Reduction Goals*

S.A. Hutchinson and A.C. Kak

Robot Vision Laboratory
School of Electrical Engineering
Purdue University
W. Lafayette, IN 47907

Abstract

Our goal in robot planning is to develop a planner which can create complete assembly plans given as input a high level description of assembly goals, geometric models of the components of the assembly, and a description of the capabilities of the work cell (including the robot and the sensory system). In this paper, we introduce SPAR, a planning system which reasons about high level operational goals, geometric goals and uncertainty-reduction goals in order to create assembly plans which consist of manipulations as well as sensory operations when appropriate. Operational planning is done using a nonlinear, constraint posting planner. Geometric planning is accomplished by constraining the execution of operations in the plan so that geometric goals are satisfied, or, if the geometric configuration of the world prevents this, by introducing new operations into the plan with the appropriate constraints. When the uncertainty in the world description exceeds that specified by the uncertainty-reduction goals, SPAR introduces either sensing operations or manipulations to reduce that uncertainty to acceptable levels. If SPAR cannot find a way to sufficiently reduce uncertainties, it does not abandon the plan. Instead, it augments the plan with sensing operations to be used to verify the execution of the action, and, when possible, posts possible error recovery plans, although at this point, the verification operations and recovery plans are predefined.

*This work was supported by the National Science Foundation under Grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems.

1. Introduction

Our goal in robot planning is to develop a planner which can create complete assembly plans given as input a high level description of the assembly goals, geometric models of the components to be assembled, and a description of the capabilities of the robotic work cell (both the robots and sensors). These plans would include both manipulations and sensory operations. Toward this end, we have developed SPAR, a planner which creates plans that satisfy operational, geometric and uncertainty-reduction goals. SPAR's approach to planning is to first create a high level plan containing actions like "pickup part-1," and then add constraints on the way the actions are executed, so that geometric goals are satisfied. It is also possible that additional high level actions will be added to the plan to satisfy geometric goals, for example, if a work piece must be repositioned so that an insertion operation can be performed. In order to satisfy uncertainty-reduction goals, SPAR examines the maximum uncertainty which might exist in its world description. If this uncertainty is too large to ensure successful execution of any action in the plan, sensing operations or manipulations are added to the plan in an attempt to reduce the uncertainty to acceptable levels. If this fails, rather than abandon the plan, SPAR adds sensing operations to verify the execution of the action, and when possible, adds precompiled recovery plans. By planning at these three levels, SPAR is able to start with a high level set of assembly goals and develop assembly plans which include geometric descriptions of the actions and sensing operations to reduce uncertainty and verify actions which might not succeed.

There are certain limitations to SPAR's planning abilities in its current implementation. First, SPAR only considers the "endpoints" of actions. Thus, if the plan calls for grasping an object and moving it to another place on the work table, SPAR will determine a set of constraints on the configuration used to grasp the object, and on the configuration used to place it on the table, but will not plan the motions required to move the manipulator from the first position to the second. Second, we have not incorporated any fine motion planning into our current planner. As a result of this, in some situations where compliant motion plans could be used to robustly perform an assembly task, SPAR will pessimistically declare that uncertainties are too great to guarantee successful assembly and that error detection sensing should be used at execution time. Currently there is research being done in our lab on compliant motion planning [16], and at some future point that work could be integrated with this planner. Finally, SPAR must know a priori about the locations and orientations of all the objects that participate in the assembly. Therefore, in order to be used in a real assembly cell, SPAR must be augmented with a sensing system capable of determining the positions of the objects to be manipulated. Such a system is currently under development in our lab and is described in [18, 19].

Much of today's planning research falls into one of two camps: the STRIPS family of planners (more generally the domain independent planners) [7, 14, 29, 34, 36], and the configuration space (C-space) planners [11, 21, 24]. Neither of these approaches to planning is capable of producing complete assembly plans from high level specifications of assembly goals. The domain independent planners are incapable of reasoning about geometric concerns and uncertainties in the work cell, while the C-space planners typically deal only with

the geometric specifications of individual actions. One system which plans geometric configurations based on a high level description of assembly goals is RAPT, described in [1, 28]. However, this system is not really a planner. Given a symbolic description of the relationships which must hold in the goal state, RAPT manipulates equations which correspond to these relationships to derive homogeneous transformations for the goal relationships between the manipulator and the work pieces to be manipulated.

Attempts at dealing with world uncertainties also fall into two camps: planners which attempt to anticipate, and avoid errors [5, 27], and error recovery planners which are invoked only after an execution error occurs [3, 15, 25, 32, 33]. Consideration of uncertainty in fine motion planning has been discussed in [6, 10, 13]. A number of schemes for representing uncertainty in robotic systems have been described, for example [12, 31], but these are not currently part of a planning system.

There are two systems of which we are aware whose scope is similar to the scope of SPAR: TWAIN [22] and Handey [23]. Each of these planners begins with a high-level task plan and then adds motion plans for the individual actions in that plan. TWAIN is a constraint posting planner which can also add sensory operations to the plan reduce uncertainties. Handey is an integrated system which includes a sensory system to determine the initial world state. One of Handey's main strengths is its ability to plan grasping operations when the objects are in cluttered environments (this is discussed further in [35]).

The approach to planning which we describe in this paper was inspired by Chapman's work on the constraint posting method [7]. In the constraint posting method, the planner seeks to satisfy a goal by first examining all of the actions and constraints previously generated to see if the goal can be satisfied by the addition of a new constraint (where a constraint may be viewed as a specification or a restriction on an action). Only if this strategy fails will a new action be added to the plan. Chapman's planner by itself would be incapable of handling the geometric and uncertainty-caused complexities in a robot work cell, and its main virtue lies in the fact that it possesses some elegant theoretical properties, such as the property of completeness which implies that if a solution to a planning problem exists, the planner would find it.

Our planner expands Chapman's constraint posting work by extending the planning beyond high-level goals (which we refer to as operational goals) to include geometric and uncertainty-reduction goals. In order to plan with these additional goals, we have added a constraint manipulation system (CMS) which contains domain-specific knowledge (including the kinematics of the robot, object models, and sensing operations). This domain knowledge is used by the CMS to determine whether or not constraints on such things as robot arm configurations can be satisfied.

When designing a constraint posting planner, the degree to which constraint posting is used is an issue which must be considered. A pure constraint posting planner makes no variable instantiations until all of its goals have been satisfied, at which time the CMS is used to determine the variable instantiations which simultaneously satisfy all of the constraints in

the constraint network. The advantage to this approach is that the planner is able to decrease the amount of backtracking by avoiding arbitrary choices which could lead to failure. The disadvantage to a pure constraint posting approach is that maintaining the constraint network can become more expensive than backtracking during planning. Therefore, in many cases a combination of constraint posting and backtracking is appropriate, the exact combination being determined by the complexities of the constraints and the cost of backtracking.

In SPAR, due to the complexities involved with the representation and evaluation of uncertainty-reduction goals, only the operational and geometric goals are satisfied using the constraint posting method (we will elaborate on these complexities in Section 4.3). Therefore, SPAR performs its planning in two phases. In the first phase constraint posting is used to construct a family of plans that satisfy all operational and geometric goals. In the second phase, specific plan instances (generated by instantiating the plan variables so that the constraint network is satisfied) are used as input for the uncertainty-reduction planning. We should note that the constraint posting paradigm is conceptually able to handle all three types of goals, however, for the reasons of complexity that we have just mentioned, it is not expedient to try and force uncertainty-reduction planning into the constraint posting mold. Furthermore, it would not be advantageous to abandon constraint posting for the operational and geometric planning, since the cost of maintaining the constraint network associated with these two types of goals is significantly less than the cost of implementing a backtracking search algorithm.

In this paper we present an overview of SPAR. A more detailed account can be found in [17]. The remainder of this paper is organized as follows: In Section 2 we will give an overview of the system, including the top-level search strategy used to satisfy system goals. In Section 3 we will describe the representations that SPAR uses for uncertainty, plans, actions, and goals. Section 4 describes how SPAR satisfies individual goals. This includes discussions on the satisfaction of high level operational goals, geometric goals, and uncertainty-reduction goals. In Section 5 we discuss how SPAR represents and manipulates constraints. Section 6 brings the first sections of the paper together by presenting an example of how SPAR develops a plan for a basic assembly task. Finally, Section 7 concludes the paper with a summary and allusions to future efforts.

2. Planning in SPAR

In order to create complete assembly plans, we have extended the planning which is done in traditional constraint posting planners, such as those described in [7, 36], to include both geometric planning and uncertainty-reduction planning. By geometric planning, we mean the planning which determines the actual geometric configurations that will be used during the assembly process. These configurations include the configurations of the manipulator, the positions in which parts are placed, and the grasping configurations which are used to manipulate objects. Uncertainty-reduction planning consists of first determining whether or not the uncertainty in the planner's description of the world (e.g. the possible error in part locations) is sufficiently small to allow plan execution to succeed. If the uncertainties are too

large, then either sensing operations or manipulations are added to the plan in an attempt to reduce the uncertainty to an acceptable level. If this fails, verification actions and local recovery plans are added to the plan. These can be used during plan execution to monitor the robot's success and recover from possible run time errors. We call the resulting planner SPAR, for Simultaneous Planner for Assembly Robots, since all three levels of planning influence one another.

In SPAR, the planning process begins with a null plan and a set of goals which the user supplies. This null plan is then refined until all goals are satisfied. This occurs in two phases. First, a constraint posting approach is used to satisfy all operational and geometric goals. Then, a second phase of planning is used to satisfy the uncertainty-reduction goals.

In the first phase of planning, SPAR iteratively refines its current partial plan so that it satisfies some pending goal. This is done by either constraining the execution of an action that is already in the plan, or by introducing a new action into the plan. In the latter case, SPAR adds the new action's geometric and operational preconditions to appropriate goal stacks, and also checks each currently satisfied goal, noting those which are possibly undone by the new action and placing them on the appropriate pending goal stack. The first phase of planning terminates when there are no more pending operational or geometric goals.

In the second phase of planning, SPAR does not use the constraint posting approach. Instead, the uncertainty-reduction preconditions are considered for specific plan instances. In order to create these plan instances, SPAR invokes its CMS to find consistent solutions for the plan's constraint network. These solutions are then used to instantiate the variables in the plan actions. Specific plan instances are examined until one is found in which all uncertainty-reduction goals can be satisfied. If no such instance can be found, the instance which contained the fewest unsatisfied uncertainty-reduction goals is selected. Furthermore, the plan instance is augmented to contain sensing verification actions and potential recovery plans for anticipated possible errors.

Figure 1 shows a block diagram of SPAR. To the left are the goal stacks and set of satisfied goals which are used to keep track of planning progress. At the top, enclosed by a dashed box, are the templates which are used to represent actions, a set of rules for instantiating those templates, a set of actions to be used to reduce uncertainty in the world, and a set of procedures which are used to construct the uncertainty-reduction preconditions for actions in the plan. To the right, enclosed by a dashed box, is the constraint system. This includes the actual CMS, a number of domain-dependent modules (e.g. upper and lower bounding routines, an algebraic simplifier, inverse kinematics of the robot), and a constraint network which is used to organize the plan's constraints. Finally, at the bottom of the figure are the verification sensory operations and local recovery plans, which are used when uncertainty-reduction goals cannot be satisfied, as well as the set of actions which are currently in the plan.

3. Representational Issues in SPAR

One of the important issues which must be addressed when designing a planning system is the choice of representation schemes which will be used. These representations determine the power that the planner will have in terms of its ability to adequately model the world and the possible actions which can be performed to alter the world. In this section, we will describe how SPAR represents actions, uncertainty, plans, and goals.

3.1. Representation of Actions

Currently, SPAR plans with three actions: pickup, putdown, and assemble.* These actions are represented by action templates, each of which has the following components.

- Action id: An identifier which SPAR uses to reference a particular instance of the action.
- Action: The name of the action, and its arguments.
- Preconditions: The operational geometric, and uncertainty-reduction preconditions which must be met prior to the execution of the action.
- Add list: A list of conditions which will be true in the world after the execution of the action.
- Delete list: A list of conditions which will no longer be true in the world after the execution of the action.

Figure 2 shows the action template for the pickup action. The meanings of the various preconditions will be made clear in the following sections of the paper.

When SPAR adds an action to the plan, it instantiates the template for the action so that it will accomplish the particular goal which caused the action's addition. This consists of first instantiating the various identifiers in the action to unique labels (e.g. the ActionId, the Gi's), and then either instantiating or constraining the plan variables in the action so that it achieves the goal. SPAR uses a set of rules to determine the proper variable instantiations for an action template, given the goal which the action is to achieve. Figure 3 shows an example of the rule which instantiates a pickup action template to achieve the goal holding(Object, Grasp). (Note that the uncertainty-reduction preconditions do not appear in the instantiated template. This is because in SPAR's current implementation, they are actually encoded as procedures.) By using this approach to instantiating action templates, SPAR is able to use a small set of generic robot operations and instantiate these to specific actions based on the objects which will be manipulated by those actions.

*There is only a limited repertoire of actions that can be carried out by a single robot arm and the three listed here represent those that are used most often. Actions like threading and fixturing could be considered to be more specialized forms of the assemble action presented here, the specialized forms being obtained by the addition of more geometric and uncertainty-reduction constraints.

In some cases, when an action is added to the plan, an initial set of constraints on the plan variables is also added. For example, the rule shown in Figure 3 specifies the initial constraint that Grasp be one of the possible grasps for Object. (We will discuss how grasps, stable poses, etc. are represented in the Section 5.) This amounts to assigning an initial label set to a node in the constraint network (this will also be discussed further in Section 5). When initial constraints are added, there is no need to invoke the CMS to see if they are consistent with the current constraint network, since the variables which will be constrained by these initial constraints did not previously exist in the plan, and thus did not occur in the constraint network.

3.2. The Representation of Uncertainty in SPAR

In order to create assembly plans which are to be executed in an uncertain environment, SPAR must have a suitable representation for the uncertainty in its world description, an understanding of how much uncertainty in that description can be tolerated before an action can no longer be guaranteed to succeed, and a knowledge of how the various assembly actions affect the uncertainty in the world description. In this section, we will address each of these three issues.

3.2.1. Representing Uncertain Quantities

In our current implementation of SPAR, we have chosen to limit the number of quantities which are considered to be uncertain. (Once the system is extended to include a sensing system, we will also consider uncertainties in object identity.) For an object resting on the work table, the X,Y,Z location of the object (i.e. the object's displacement) and the rotation about an axis through the origin of the object's local frame and perpendicular to the table are considered uncertain. This choice reflects our assumption that objects resting on the work table will be in a stable pose, which fixes two rotational degrees of freedom of the object (this assumption will be discussed further in Section 5). For the manipulator, we consider the X, Y, Z location of the tool center and the rotation about the Z axis of the manipulator's local frame to be uncertain.

All uncertainties in SPAR are expressed in terms of uncertainty variables. The possible values for an uncertainty variable are defined using bounded sets. We represent the uncertainty in the position of an object by a homogeneous transformation matrix whose entries are expressed in terms of uncertainty variables. By combining the ideal position of an object (i.e. the position of the object if all uncertainty is eliminated) with the uncertainty in that position, we obtain the possible position of an object. This possible position will be a homogeneous transformation matrix, with some or all of its entries expressed in terms of uncertainty variables. Any matrix which can be obtained by substituting valid values for the uncertainty variables will represent one possible position of the object.

Given these assumptions, we define the transformation which represents the uncertainty in the position of the manipulator relative to the manipulator's local frame to be:

$$T_{\Delta M} = \begin{bmatrix} \cos(\Delta\theta_g) & -\sin(\Delta\theta_g) & 0 & \Delta X_g \\ \sin(\Delta\theta_g) & \cos(\Delta\theta_g) & 0 & \Delta Y_g \\ 0 & 0 & 1 & \Delta Z_g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Again, note that the values, ΔX_g, ΔY_g, ΔZ_g, and $\Delta\theta_g$ are bounded symbolic variables. Therefore, the matrix $T_{\Delta M}$ represents all of the transformations which could be obtained by substituting valid numerical values into the matrix in place of the symbolic variables. The bounds on these variables are stored in SPAR's database, and retrieved when needed.

Given that $T_{\Delta M}$ represents the uncertainty in the manipulator's position relative to the manipulator's own local frame, we can compute the possible position of the manipulator (i.e. the combination of ideal position and possible error) using the composition:

$$T_{M+\Delta} = T_M \, T_{\Delta M} \tag{1}$$

where T_M represents the ideal position of the manipulator.

The expression for the possible position of an object resting on the work table is a bit more complicated, due to the rotational component in the uncertainty. In particular, the axis of this rotation is not defined by the local frame of the object or by the world frame, but by the world Z axis, translated to the origin of the object's local frame. To deal with this, we will consider the uncertainty in the displacement of the object and the rotational uncertainty separately. For the displacement, let the transformation $Tr_{\Delta O}$ be a transformation which defines the uncertainty in the X,Y,Z location of the object relative to the world coordinate frame:

$$Tr_{\Delta O} = \begin{bmatrix} 1 & 0 & 0 & \Delta X_O \\ 0 & 1 & 0 & \Delta Y_O \\ 0 & 0 & 1 & \Delta Z_O \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, let the transformation Tr_O represent the ideal X,Y,Z position of the object. We obtain the possible displacement of the object's local frame by combining the two:

$$Tr_{O+\Delta} = Tr_{\Delta O} \, Tr_O$$

Now, since the rotational uncertainty is about the world Z axis translated to the origin of the object's local frame, it can be represented by post-multiplying the possible object displacement by a rotation about the Z axis, $R_{\Delta O}$ where:

$$R_{\Delta O} = \begin{bmatrix} \cos(\Delta\theta_O) & -\sin(\Delta\theta_O) & 0 & 0 \\ \sin(\Delta\theta_O) & \cos(\Delta\theta_O) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, by defining the matrix R_O to denote the ideal orientation of the object, we obtain the possible position of the object (which includes both displacement and rotation uncertainties) as:

$$T_{O+\Delta} = Tr_{\Delta O}\, Tr_O\, R_{\Delta O}\, R_O \tag{2}$$

3.2.2. Derivation of Uncertainty-Reduction Goals

In order to illustrate the construction of the uncertainty-reduction preconditions, in this section we will derive the uncertainty-reduction preconditions for the pickup action. This is done by examining the possible locations of the manipulator fingers, and their relationship to the possible locations of the contact points on the object to be grasped (i.e. the points on the object which the fingers will contact in the grasping operation). These preconditions should guarantee that the two contact points will lie between the fingers of the manipulator, even when worst case uncertainties occur. To express this, we first derive the possible local coordinate frames for each finger. We then derive the possible locations of the contact points on the object. Finally, we transform the possible contact points so that they are expressed in the local finger frames, and check that they each lie between the fingers.

To find the possible local frames of the manipulator fingers, we find the possible location of the manipulator's local frame and perform a translation of $\pm 1/2W_m$ along the Y axis of that frame (where W_m is the distance between the two fingers). This is illustrated in Figure 4. Using Equation 1, we find:

$$P_1 = T_{M+\Delta}\, trans(0, -1/2W_m, 0)$$

$$P_2 = T_{M+\Delta}\, trans(0, +1/2W_m, 0)$$

where trans (X, Y, Z) indicates a transformation of the form:

$$trans(X, Y, Z) = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this case, the possible position of the manipulator is obtained by using Equation 1, but replacing T_M by the composition of the object position (T_O) and the grasping configuration, T_G (which expresses the position of the manipulator's coordinate frame relative to the object's local frame).

In order to determine the two possible contact points, C_1 and C_2, we first find the possible position of the object. Relative to the object's possible local frame, the contact points are obtained using the grasping transformation, T_G, in conjunction with a translation along the Y axis of the manipulator frame (i.e., the axis which defines the direction of finger opening and closing). Using Equations 2, we find:

$$C_1 = Tr_{\Delta O}\, Tr_O\, R_{\Delta O}\, R_O\, T_G\, trans(0, +1/2W_g, 0)\, [0, 0, 0, 1]^t$$

$$C_2 = Tr_{\Delta O}\, Tr_O\, R_{\Delta O}\, R_O\, T_G\, trans(0, -1/2W_g, 0)\, [0, 0, 0, 1]^t$$

where W_G is the width of the object at the grasp point. Note that we are not interested in the coordinate axes at the contact points, only the displacement.

In order to see if the contact points lie between the fingers, we transform the locations of C_1 and C_2 to be defined in terms of the coordinate frames P_1 and P_2, and check to see that the Y-components of these locations are on the positive Y axis for P_1 and on the negative Y axis for P_2 for all possible values of the uncertainty variables. Therefore, the four uncertainty-reduction preconditions for the pickup action are:

$$0 < [0, 1, 0, 0]\, P_1^{-1}\, C_1,\ 0 < [0, 1, 0, 0]\, P_1^{-1}\, C_2$$

and

$$0 > [0, 1, 0, 0]\, P_2^{-1}\, C_1,\ 0 > [0, 1, 0, 0]\, P_2^{-1}\, C_2$$

Again, note that all of the matrix multiplications shown above must be performed symbolically. This is because many of the entries in the matrices will be expressed in terms of uncertainty variables which do not have specific numeric values.

3.2.3. The Propagation of Uncertainty by Actions

To illustrate how actions propagate uncertainty, in this section we will describe how the pickup action affects the uncertainties in the position of the object to be grasped. In general, the pickup action has the effect of reducing the uncertainty in the position of the object to be grasped. This is because the new uncertainty in the object's position will be defined in terms of the manipulator uncertainty, which is normally less than the uncertainty in positions which are determined by the sensing system. Specifically, the pickup action has the effect of transforming the object's displacement uncertainty into the manipulator coordinate frame, and then reducing the Y component of this uncertainty to the uncertainty in the Y component of the manipulator. The pickup action also reduces the uncertainty in the object's orientation to be equal to the uncertainty in the orientation of the manipulator.

In order to represent this, let $Tr_{\Delta O}$ be the displacement uncertainty in the object's position just prior to the execution of the pickup action. Therefore, as described in Section 3.2.1, this uncertainty is defined relative to the world coordinate frame. We need to obtain a displacement error $Tr'_{\Delta O}$ such that

$$R_M \, Tr'_{\Delta O} = Tr_{\Delta O}$$

where R_M is the transformation which represents only the orientation of the manipulator (i.e. it has a null displacement vector). In other words, $Tr'_{\Delta O}$ expresses the displacement error relative to the manipulator frame after the object is grasped. If we define R_O to be the transformation that represents the orientation of the object, and R_G to be the transformation which represents the orientation of the manipulator relative to the local frame of the object (i.e. the rotational part of T_G), we find:

$$Tr'_{\Delta O} = (R_O \, R_G)^{-1} \, Tr_{\Delta O}$$

given that

$$R_M = R_O \, R_G$$

and therefore,

$$R_M \, [(R_O \, R_G)^{-1} \, Tr_{\Delta O}] = Tr_{\Delta O}$$

Now, we define the vector that represents the uncertainty in the object's displacement relative to the manipulator frame by:

$$[Dx, Dy, Dz, 1]^t = (R_O \, R_G)^{-1} \, Tr_{\Delta O} [0, 0, 0, 1]^t$$

Finally, by combining this displacement with the uncertainty in the position of the manipulator, we obtain:

$$T_{\Delta O} = \begin{bmatrix} \cos(\Delta\theta_g) & \sin(\Delta\theta_g) & 0 & Dx + \Delta X_g \\ \sin(\Delta\theta_g) & \cos(\Delta\theta_g) & 0 & \Delta Y_g \\ 0 & 0 & 1 & Dz + \Delta Z_g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that the uncertainty in the Y component of the displacement uncertainty has been limited to the uncertainty in the Y component of the location of the manipulator's tool center. Further, note that the rotational uncertainty is the same as the rotational uncertainty in the orientation of the manipulator.

3.3. Representation of Plans

As described earlier, SPAR does its planning in two phases. During the first phase, a constraint posting approach is used to satisfy operational and geometric goals, and during the second phase, specific plan instances are examined to find plans which satisfy uncertainty-reduction goals. Clearly, the representation of plans must be different for these two phases.

The plans developed by SPAR during the first phase of planning are not simple linear sequences of actions. Instead, these plans consist of an unordered set of actions and a separate set of constraints on how and when those actions are to be executed. These constraints are stored in SPAR's constraint network. The constraints on how actions are executed are actually constraints on possible values which may be assigned to plan variables. For example, if the action is to grasp an object, constraints on the variable used to indicate the grasping configuration will effectively constrain how the grasping action is performed. Table 1 lists the constraints SPAR currently uses in the first phase of planning.

With this type of representation, a plan developed by SPAR during the first phase of planning actually corresponds to a family of plans. A specific plan instance is derived by finding a consistent instantiation for the plan variables (i.e. a set of values for the plan variables which satisfies the constraints in the constraint network) and performing that instantiation on the plan actions.

In the second phase of planning, SPAR uses specific plan instances, which are augmented to contain verification sensory actions, local recovery plans, and an error count. The verification-sensory actions and local recovery plans are added when the uncertainty reduction preconditions for an action cannot be satisfied. The error count is incremented each time the uncertainty-reduction goals for an action in the plan instance cannot be satisfied. This error count is used to determine which plan instance has the greatest chance of success.

3.4. Representation of Goals

In this section, we describe SPAR's representation of goals. Since Section 3.2 dealt with SPAR's representation of uncertainty and uncertainty-reduction goals, we will not discuss the uncertainty-reduction preconditions of actions in this section.

Goals in SPAR have three relevant attributes: a type (either operational, geometric, or uncertainty-reduction), a condition which must be satisfied (i.e. the actual goal), and an action identifier. The action identifier is used to indicate when the goal must be satisfied, in particular, that it must be satisfied prior to the execution of the action specified by the action identifier. We will use the terms goal and precondition to refer to either the condition part of the goal or to the entire structure. Which of these is meant should be clear by the context.

SPAR's operational goals are similar to the high-level goals used in traditional domain independent planners (e.g. STRIPS or TWEAK). One difference is our inclusion of plan variables which can be used to link the operational and geometric goals. For example, one operational precondition of the assemble action is:

op(G1, ActionId, holding(Obj1, Grasp))

The plan variable Grasp is not used in the operational planning, but serves the purpose of linking the operational and geometric planning. The variable ActionId is used to indicate the time at which the goal must be satisfied. In particular, it must be satisfied just prior to the execution of the action whose action identifier is ActionId.

Geometric goals are slightly more complex, with two main components. The first is a geometric constraint and the second is a set of operational goals. The meaning of this pair is that the planner is to establish the operational goals in such a way that the geometric constraint is satisfied. For example, one geometric precondition of the putdown action is:

```
geo(G2, ActionId,
    reachable(Grasp, Pos),
    holding(Obj, Grasp))
```

4. Goal Satisfaction

In this section, we will individually discuss the methods that SPAR uses to satisfy operational, geometric, and uncertainty-reduction goals. In the course of this discussion, we will frequently allude to the CMS's role in the process of goal satisfaction; however, we will leave a detailed discussion of the CMS for Section 5. For the purposes of this section, it is sufficient to assume that the CMS is capable of determining if a new constraint is consistent with the current constraint set.

4.1. Satisfying Operational Goals

In SPAR, ensuring the satisfaction of an operational goal proceeds in two steps: finding an action which establishes the goal and then dealing with actions that could clobber (or undo) the goal.

In order to find an action which establishes an operational goal, SPAR first looks at the add lists of the actions that are already in the partially developed plan. If any element of the add list of such an action can be unified with the operational goal, then that unification is performed and the action is declared to have established the goal. If SPAR succeeds in finding such an action, that action is constrained to take place prior to the time that the goal must be satisfied. If the CMS determines that this new ordering constraint is not consistent with the current constraint network, the constraint addition fails and SPAR backtracks in an attempt to find another action in the plan which establishes the goal.

If SPAR fails to find an action in the plan that can establish the goal, it adds a new action. This consists of instantiating an action template, adding the action to the plan, and constraining the new action to occur prior to the time that the goal must be satisfied. Any time SPAR adds an action to the plan, it is possible that the new action may clobber goals which

have already been satisfied. For this reason, when a new action is added to the plan, SPAR examines the list of satisfied goals and transfers any of these which could be clobbered by the new action to the appropriate pending goal stack.

Once an operational goal has been established, SPAR examines each action in the current partial plan to see if it could possibly clobber the goal. An action can clobber an operational goal if any element in the action's delete list can be unified with the goal. There are three ways to deal with a potential clobberer. The clobbering action can be constrained to occur after the time that the goal must be satisfied (promotion of the goal). The goal can be constrained not to unify with the clobbering clause in the action's delete list (separation). An action can be used to re-establish the goal (white knight). This white knight can either be an action that is already in the plan, or it can be a new action that is added specifically for the purpose of re-establishing the clobbered goal.

In SPAR, since pattern matching is done using unification, it is difficult to add separation constraints to the plan. The reason for this is that the unification is done using Prolog's unification algorithm, which will not take into account constraints in SPAR's constraint network. Therefore, it is difficult to implement a constraint which says that an element in the delete list of an action, for example holding(part1, Grasp), should not be instantiated so that it matches a particular goal, for example holding(part1, grasp1). For this reason, we have omitted separation as a possible means of declobbering goals in SPAR.

Promotion of the goal is the first option that SPAR tries when declobbering goals. When an action, C, can clobber a goal required to be true during the execution of a certain action, S, SPAR attempts to add a constraint of the form prior_to(S,C), which specifies that the potential clobberer should not be executed until after action S has been executed. If this constraint addition fails, SPAR will attempt to remedy the possible clobbering by the addition of a white knight.

Using a white knight to re-establish a goal is the same as establishing a goal, with the additional condition that the white knight must occur after the potential clobberer. As such, this process proceeds exactly as the establishment process described above, but when a candidate action is found, the additional constraint prior_to(C,W) is added to the constraint network (where C is the action identifier of the clobberer and W is the action identifier of the white knight). In an earlier paragraph we mentioned the possibility of constraining the clobberer to occur before the establishing action. This is the same as following the establishing action to act as its own white knight.

4.2. Satisfying Geometric Goals

In SPAR, geometric goals are satisfied by constraining the way that plan actions are performed. For example, if a geometric goal specifies that the manipulator should be holding an object in a particular grasping configuration, the way to satisfy that goal is to place a constraint on how the manipulator performs the grasping action. In order to do this, SPAR needs to link together the operational and geometric levels of planning. For this purpose, when planning to satisfy operational goals, plan variables are introduced which can be

constrained by the geometric level of planning to determine how an action is executed. The geometric preconditions are expressed in terms of those variables. For example, a traditional STRIPS type action is pickup(Object). SPAR's equivalent action is pickup(Object, Grasp). The variable Grasp is used to define the geometric configuration which will be used by the manipulator in grasping the object. At the operational level, the variable Grasp is primarily ignored, but its presence gives SPAR a method of constraining how the pickup operation is actually performed, thus linking distinct levels of planning.

As we pointed out in Section 3.4, geometric goals consist of a set of operational goals and a geometric constraint which is to be applied to the actions that achieve the operational goals. Each operational goal that is associated with a geometric precondition of an action is also listed separately as an operational precondition of the action. Therefore, since SPAR only considers geometric goals when the operational goal stack is empty, the operational goals associated with a geometric goal are guaranteed to be satisfied by the current partial plan. Therefore, in order to satisfy a geometric goal SPAR first finds the actions which establish its associated operational goals, and attempts to constrain the execution of those actions so that the geometric constraint is satisfied. This is done by instructing the CMS to add the geometric constraint to the constraint network. If this succeeds, the goal is satisfied and moved to the list of satisfied goals.

If the CMS determines that the geometric constraint is not consistent with the current constraint network, then one or more new actions must be added to the plan. These new actions are chosen based on the operational goals associated with the geometric goal. The instantiation of the actions' templates proceeds as described in Section 3.1. Once the actions have been added, the appropriate geometric constraint is also added to the constraint network. This constraint will automatically be consistent with the constraint network, since the new action will contain new plan variables which have not yet been constrained. Note that the addition of actions to the plan will introduce new operational goals, and therefore effectively transfer control back to operational planning.

There is no need for SPAR to check for actions that might clobber geometric constraints. The reason for this is that the constraint network has no sense of temporal ordering. The entire network must be consistent at all times. Therefore, if any constraint in the network had the effect of clobbering the new geometric constraint, this would have been detected by the CMS when attempting the constraint addition.

4.3. Satisfying Uncertainty-Reduction Goals

When there are no remaining operational or geometric goals, SPAR begins the second phase of planning, which deals with uncertainty-reduction goals. There are two fundamental differences between this phase and the first phase of planning. First, the uncertainty-reduction planning does not use the constraint posting method. Second, if no plan instance can be found which satisfies all uncertainty-reduction goals, SPAR does not backtrack to the geometric and operational levels of planning. Instead, it prepares for possible failures by adding verification steps and potential local recovery plans.

As we mentioned earlier, we do not use constraint posting to satisfy uncertainty-reduction goals due to the complexities involved with their representation and evaluation. The high cost of representing uncertainty-reduction goals compared to either operational or geometric goals is partially due to the fact that the geometric and operational effects of actions do not propagate through more than one action, but uncertainties may propagate through many actions. For example, consider the sequence of actions:

action1:pickup(part1,grasp1)
action2:putdown(part1,position1)
action3:pickup(part1,grasp2)

After the execution of action2, part1 will be in a particular position (which is represented by the variable position1) regardless of where it was prior to the execution of action1. However, the uncertainty in the location of part1 after the execution of action2 will be a function of many variables, including the uncertainty in the position of the manipulator during the execution of action1 and action2, how the particular grasping configuration used in action1 affects the uncertainty in the location of part1, and the uncertainty in the location of part1 prior to the execution of action1. Therefore, while the geometric preconditions of action3 can be expressed in terms of two plan variables (position1 and grasp2), the uncertainty preconditions depend on every action prior to action3 which involved part1.

The fact that uncertainties can propagate through an indefinite number of actions also affects the complexity of evaluating the uncertainty-reduction constraints. As described in Section 3.2, SPAR uses symbolic algebraic expressions to represent uncertainty. Each time a plan action affects the uncertainty in some quantity, there is, in the worst case, a multiplicative increase in the number of terms in the corresponding symbolic expressions. Therefore, the number of terms in an expression for an uncertain quantity is, in the worst case exponential in the number of actions in the plan. Since the CMS uses upper and lower bounding routines to evaluate uncertainty-reduction constraints, and since the time complexity of these routines is a function on the number of terms in the input expression, the worst case time complexity for the evaluation of an uncertainty-reduction constraint is exponential in the number of actions in the plan. In contrast, constraints associated with operational and geometric goals can, in the worst case, be evaluated in time that is polynomial in the number of actions in the plan. In the best case, the time is constant (e.g. in evaluating constraints on the robot's joint angles).

There are two reasons for not backtracking into the first phase of planning. Firsts, since SPAR represents uncertainty in the world using bounded sets (e.g. the X location of an object would be represented as $X \pm \Delta X$), even though uncertainty-reduction goals cannot be satisfied, it is quite possible that the actual errors in the world description will be small enough that the plan can be executed without failure. Therefore, SPAR adds verification sensory actions and local recovery plans to offending plan instances, in anticipation of possible execution error. Second, by using the constraint posting approach in the first phase

of planning, SPAR attempts to develop the most general plan which will satisfy the operational and geometric goals. Therefore, it is not likely that a great deal could be gained by backtracking into the first phase of planning.

The top level of uncertainty-reduction planning consists of a loop in which specific plan instances are generated and tested until one is found in which all uncertainty-reduction goals can be satisfied. If all possible plan instances have been generated and none are without violated uncertainty-reduction goals, the instance with the fewest violations is selected for execution.

The uncertainty-reduction planning for a particular plan instance begins with the creation of an augmented-plan instance which contains four components: the instantiated list of plan actions (obtained by instantiating the actions from the partial plan that was developed in the first phase of planning so that all constraints in the constraint network are satisfied), an error count (initially set to zero), a list of sensory-verification actions (initially set to the empty list), and a list of local error recovery plans (also initially set to the empty list). Once this augmented plan instance has been constructed, SPAR sequentially examines each individual action in the instantiated action list and attempts to satisfy its uncertainty-reduction preconditions. After an action has been considered, its add and delete lists are used to update the world state to reflect the effects of the action. This has the effect of propagating the uncertainty in the world description forward, thereby defining the uncertainty in the world when the next action in the sequence will be executed.

The first step in satisfying an uncertainty-reduction goal for an individual action is the construction of the symbolic algebraic inequality associated with that goal. This is achieved by performing an appropriate combination of symbolic matrix multiplications, matrix inversions, etc., as determined by the actual goal. It should be noted that many of the quantities which enter into these operations will be defined in the world state (e.g. the part locations, uncertainties in the part locations).

If the uncertainty in the world description exceeds that which is specified by an uncertainty-reduction goal, SPAR introduces sensing operations into the plan in an attempt to reduce the offending uncertainties. Sensing actions have the same representation as manipulations. The add and delete lists of a sensing action template contain elements which describe how the uncertainties in the world description are reduced by the action. Once the sensing actions have been inserted into the plan instance, these add and delete lists are used to update the world state. The resulting world state is then used to recompute the symbolic expressions for the failed uncertainty-reduction goal.

If the sensing operations fail to reduce the uncertainty to acceptable levels, SPAR attempts to introduce manipulations into the plan which can reduce the uncertainty. Currently, the only manipulation which is used for this purpose is squeezing an object between the manipulator fingers. The reduction in uncertainty for this action is the same as the reduction in uncertainty for the pickup action, as was described in Section 3.2.3. Since the operational and geometric preconditions for this action are the same as for the pickup action,

it can always be spliced into the plan instance just prior to the execution of some existing pickup action; however, the uncertainty in the world description must satisfy the uncertainty-reduction preconditions for the uncertainty-reduction pickup action.

If the sensing operations and manipulations fail to sufficiently reduce uncertainties, SPAR prepares for possible execution error. First, the error count for the augmented plan instance is incremented by one. Second, a sensing verification action and a local recovery plan are added to their respective lists in the augmented plan instance. We should point out that the process of instantiating verification strategies and local recovery plans is in its formative stages. At this point, methods tend to be ad hoc, based on the programmer's evaluation of possible errors and likely recovery plans. We hope that future work will enable us to link CAD modeling systems with SPAR's descriptions of worst case world error to automatically predict the types of errors which could occur, and automatically prescribe verification strategies and recovery plans.

5. Constraint Manipulation

In SPAR, the bulk of the domain knowledge resides in the constraint manipulation system. This allows the top-level planning to proceed without any need to "understand" the domain of automated assembly. The actions include preconditions on the geometry of actions and the tolerable uncertainties in the world description, but in order to satisfy these preconditions, the top level planner merely requests that the CMS add constraints to the constraint database. It is the task of the CMS to determine whether or not these new constraints are consistent with the current constraints in the plan, which in turn, requires a certain amount of domain-specific knowledge.

SPAR currently uses three types of constraint. In operational planning, SPAR uses ordering constraints to ensure that actions are performed in the proper sequence (and that goals are satisfied at the appropriate times). In geometric planning, SPAR uses binary constraints between object positions and manipulator configurations to ensure that the robot will be able to perform the required manipulations. Finally, at the uncertainty-reduction level, symbolic algebraic inequalities are used to express the maximum uncertainty which can exist in the world description prior to the execution of an action.

Throughout the previous sections of the paper, we referred to the CMS maintaining a constraint network. In actuality, there is not a single, uniform constraint network. A directed graph is used for ordering constraints, a binary constraint network is used for the geometric constraints, and algebraic inequalities (expressed in terms of bounded symbolic variables) are used for the uncertainty-reduction constraints. This separation does not interfere with determining the consistency of the constraint set, since the three types of constraints do not interact. For example, even though operational planning might influence the choice of which geometric constraint to add in the course of satisfying a particular geometric goal, once that geometric constraint is chosen, it will be expressed solely in terms of geometric quantities. Therefore, in the constraint database, there will be no interaction between distinct types of constraints.

In this section, we will describe the constraints that are used in SPAR, their semantics, and how the CMS determines whether or not new constraints are consistent with the current constraint set. At this time, SPAR's CMS is not complete in the sense that it is possible that a new constraint will be determined to be inconsistent when it really isn't. The reason for this is that the quantities which enter into the constraints in SPAR are very complex and often, exact solutions are only approximated. For example, characterizing the space of reachable grasps for a robot entails partitioning a six-dimensional space into reachable and unreachable regions. In SPAR, we have devised a representation of grasping configurations which approximates the true situation. This simplifies the process of constraint manipulation, but adds the possibility that SPAR might overlook certain solutions.

5.1. Ordering Constraints in Operational Planning

In the first phase of planning (used to satisfy operational and geometric goals), SPAR operates as a nonlinear planner, so there is not a total ordering of the actions in the plan. Instead, the time of an action's execution is specified by a set of ordering constraints. Each such constraint specifies whether the action should be executed before or after some other action in the plan. While it is possible that the set of ordering constraints in a plan will define a total ordering of the plan steps, more often it will only define a partial ordering.

SPAR's CMS uses a directed graph (which we will refer to as the ordering graph) to keep track of ordering constraints. All actions in the plan are represented in the ordering graph. Any time a new action is added to the plan, a new node is created in the ordering graph, with the action's action identifier as the node's label. An ordering constraint of the form prior_to(Action1,Action2) is represented by an arc directed from the node for Action1 to the node for Action2. Consistency of the ordering constraints is guaranteed as long as the ordering graph contains no cycles, since the only type of inconsistency which might arise is if an action is constrained to occur both prior to, and also after some other action in the plan.

5.2. Constraints at the Geometric Level of Planning

All of the geometric constraints in SPAR are either binary constraints between plan variables representing object positions and manipulator positions, or unary constraints on plan variables. Furthermore, both object poses (i.e. possible orientations of objects, not including displacement information) and grasping configurations have been quantized, and assigned labels, so that each of these can be represented by a single symbolic variable rather than a continuous variable in six-dimensional space. Because of these qualities, it is straightforward to represent the geometric constaints using a binary constraint network. By using a binary constraint network, when the CMS is instructed to add a new constraint, the consistency of that constraint with the current set of constraints can be determined by adding an arc to the constraint network and then checking the new network for consistency.

We will begin this section with an introduction to binary constraint networks and an explanation of how such a network is used to represent SPAR's geometric constraints. Following this, we individually describe each type of geometric constraint included in SPAR, and the mechanisms used to evaluate those constraints.

5.2.1. The Geometric Binary Constraint Network

This section includes a cursory introduction to constraint networks. A more thorough introduction can be found in either of [8,9]. We begin our discussion with the following definitions.

Def: The label set for a plan variable is the set of possible values which may be assigned to that variable.

Def: A unary constraint on a variable is a restriction of that variable's label set.

Def: A binary constraint on two variables, V_i and V_j, is a relation

$$C_{ij} \subset L_i \times L_j$$

where L_i is the label set of V_i and L_j is the label set of V_j.

Def: A binary constraint network is an undirected graph whose nodes represent constrained variables, and whose arcs represent constraints between variables.

SPAR's CMS uses depth first search with backtracking to determine network consistency. For each level in the search, this consists of selecting one node in the network which has not yet been assigned a value, and assigning to it a value which is consistent with all assignments that have previously been made in the search (note that for the first node, there will have been no previous assignments, and so any value from the node's label set may be chosen). The algorithm is similar to that described in [9].

In order to represent SPAR's geometric constraints using a binary constraint network, each geometric plan variable (e.g. grasp configurations, positions) is represented by a node in the network. When a new variable is introduced into the plan, a node is added to the network and assigned an initial label set. This label set is merely the set of values which may be assigned to that variable (determined by the action template instantiation rules discussed in Section 3.1). For example, if the variable represents a grasping configuration for a particular object, then the initial label set for its node in the constraint network will contain the labels of all the grasping configurations for the object (grasping configurations will be described in Section 5.2.4).

Binary constraints between plan variables are represented by arcs between the corresponding nodes in the network (these arcs are not directed). Each arc in the network contains a set of pairs of values which indicate the valid pairs of labels for the connected nodes. Determining the valid pairs of labels requires a semantic understanding of the domain, but once the pairs have been assigned, no domain knowledge is required to check for network consistency.

When the CMS is instructed to add a unary constraint to the network, it first updates the label set of the appropriate node, and then updates each arc connected to that node by deleting pairs which are no longer valid given the node's new label set. Finally, the new

network is checked for consistency. When the CMS is instructed to add a new binary constraint to the network, it adds an arc between the appropriate nodes (creating the nodes if they do not already exist in the network), and then checks for network consistency.

5.2.2. Set Membership

In order to restrict the label set of a plan variable, SPAR uses the constraint member (Variable, Labels). If there is no node in the geometric constraint network for Variable, the CMS adds one, and assigns its initial label set to contain the elements of Labels. If there is already a node in the constraint network for Variable, the CMS takes two steps to ensure that the new constraint on Variable's label set will not result in an inconsistent network. The first step ensures node consistency (i.e. that the node for Variable will have at least one possible label), and the second ensures network consistency. To ensure node consistency, the set Labels is intersected with Variable's current label set. If the intersection is empty, then the new member constraint is not consistent with the current constraint set. If the intersection is not empty, then it is assigned as Variable's new label set. To ensure network consistency, all arcs leaving the node corresponding to Variable are updated by deleting pairs that assign Variable a value which is not in its new label set. The new network is then checked for consistency. If both node and network consistency are satisfied, the CMS returns success. If not, failure is returned.

5.2.3. Stable Poses and Position Classes

For the purpose of assembly operations, the exact position of an object is not always important. What is important is that the object be oriented in a way that allows the mating features of the object to be accessible. For example, if the assembly operation is to inset a peg into a hole in a block, it is not important how the block is oriented, as long as the hole is positioned so that the peg can be inserted. In light of this, in SPAR, we characterize object positions using equivalence classes. These classes are based on the object's stable poses, where by stable pose we mean an orientation of the object which allows it to rest naturally on the work table. For example, a cube has six stable poses.

The use of stable poses to quantize the space of object positions serves two purposes. First, it provides a method for easily determining which of an object's features will be obscured by the work table. Second, when the plan calls for an object to be placed in some position (by the putdown action), most often the displacement of the object is not important. Stable poses provide a method of specifying destination positions in terms of the object's orientation, without regard to the actual X,Y,Z position. Clearly, in a cluttered work cell, objects will not always be found in one of their stable poses. However, since stable poses are only used to determine a list of occluded features and to specify destinations of held objects, this will not be a problem as long as the sensory system is capable of determining by inspection the object's occluded features.

Using this representation for object positions, geometric goals about object locations can be expressed in terms of set membership. That is, the planner can determine the set of stable poses which are allowable for a certain assembly operation and constrain the object's position to correspond to one of those poses. For this purpose, SPAR uses the constraint

in_position_class(Position, Plist). This constraint indicates that the orientation specified by Position must correspond to one of the stable poses in Plist.

If Position is instantiated to a homogeneous transformation which represents both the orientation and displacement of an object (for instance, if the object has been ascertained by the sensing system), then this constraint cannot be evaluated by a simple membership test. In this case, the CMS must determine to which stable pose of the object Position corresponds. This can be done in one of two ways. If the object is resting on the table, it is a simple matter to compare the rotational component of Position to the rotations specified by the various stable poses of the object to determine in which stable pose the object is resting. If the object is not resting on the table (e.g. if it is leaning against some other object in the work cell), then the sensing system must be used to determine the set of object features which are occluded. Position is then determined to correspond to the stable pose which obscures the same set of features.

Aside from the situation described in the last paragraph, the CMS handles the addition of an in_position_class constraint in the same way that it handles the member constraint. It restricts Position's label set, updates the arcs which are connected to Position's node, and then checks for network consistency.

5.2.4. Reachability of Grasps

Whenever the planner inserts a manipulation action into the plan, it must ensure that all of the configurations required to perform that manipulation will be physically realizable. In order to do this, SPAR uses two constraints:

reachable(Grasp,Position)

and

mate_reachable(Grasp,Position,T_a)

The first of these indicates that if the object to be manipulated is in the position specified by the variable Position, and the configuration used to grasp the object is specified by Grasp, then that combination must be physically realizable. This constraint is used both in grasping, and in placing objects. The second constraint is used for mating operations, where T_a is a homogeneous transformation which represents the destination position of the grasped object relative to the coordinate frame specified by Position.

For specific values of Grasp and Position, two conditions must be met in order for the reachable constraint to be satisfied:

1. The faces of the grasped object which come into contact with the manipulator fingers must not be in contact with the table (or any other object) when the object is located in Position.

2. The robot must be able to perform the grasp without exceeding any of its physical joint limits.

For the mate_reachable condition, only the second condition is used. However, the position which the manipulator must reach is not Position, as in the reachable constraint, but $T_O\ T_a$, where T_O is the homogeneous transformation corresponding to Position.

To verify condition 1, the system must invoke the object modeling system to determine which features of the object will be in contact with the table when the object is in Position, and which features of the object will be in contact with the manipulator when the object is grasped in the configuration specified by Grasp. (We should note that the modeling system used in SPAR is not a CSG modeler. A number of object representations are included in an object model, including a grasping model, a table of the stable poses, and a great deal of geometric information which is used by the sensing system for object recognition and localization.) If Position corresponds to one of the object's stable poses, a simple table lookup operation is used to determine which features are in contact with the table. If Position is an absolute position, then the system must determine the set of occluded features as was discussed in Section 5.2.3.

Condition 2 is verified, for specific values of Grasp and Position, by invoking routines which compute the inverse kinematic solution for the robot's joint angles given an absolute position of the end effector. This is done as follows. A particular grasp has associated with it a homogeneous transformation which defines the coordinate frame of the robot manipulator relative to the frame of the object. We will refer to this as the grasp transformation, or alternatively T_g. If Position is an absolute position (i.e. it has a specific X,Y,Z location as well as a specified orientation) specified by the homogeneous transformation T_O, we compute T, the transformation representing the manipulator's coordinate frame relative to the world frame, by $T=T_O\ T_g$. In the mate_reachable case, $T=T_O\ T_a\ T_g$. This transformation is used as the input to the inverse kinematics program. The joint angles which are found by this program are then tested to ensure that they are within the limits attainable by the robot. Currently, our lab is using a PUMA 762 robot for manipulation experiments. Descriptions of the kinematic and inverse kinematic solutions for this type of robot can be found in [20].

If Position corresponds to a stable pose (that is, it specifies an orientation of the object, but no absolute X,Y,Z position) the CMS assumes that condition 2 can be satisfied by some suitable choice of X,Y,Z. That is, we assume that for any arbitrary orientation of the robot manipulator, there will be some location in the work space where this orientation can be physically performed (where by orientation, we mean that the coordinate frame for the grasp has axes whose origin is not specified, but whose orientation relative to the world frame is specified).

When the CMS is instructed to add either a reachable or mate_reachable constraint to the constraint network, the two conditions described above are used to determine all valid pairs of values for Grasp and Position (note that T_a will always be instantiated to a constant homogeneous transformation). This is done by exhaustively pairing every value from the label set for Grasp with every value from the label set for Position, and recording all pairs which satisfy the two conditions. These pairs are then used to construct a new arc connecting

the nodes for Grasp and Position. Finally, a network consistency check is performed. If the consistency check fails, the CMS signals failure and the old network is restored. Otherwise, the CMS signals success and retains the new network.

Exhaustive enumeration of pairs of positions and grasps is not as difficult as it might seem. First, as we have described earlier, there are a finite number of possible stable poses associate with any object (if the object is in a known location determined by the sensing system, then there is only one position to consider). Usually this number is fairly small. Second, we quantize the space of grasping operations based on the features of the object which are obscured by the grasp and the features of the object which come into contact with the manipulator fingers in the grasp. This approach is similar to that described in [26, 35].

By making this type of quantization of the space of grasping configurations, we replace exact descriptions of grasping configurations with approximations. Because of this, it is possible that SPAR will occasionally determine that a reachable constraint is not consistent with the current constraint database, when in fact it is consistent. In general, we do not expect this to happen except when the manipulations which are to be performed require the robot to operate near the boundaries of its work envelope.

5.3. Constraints at the Uncertainty-Reduction Level of Planning

As we described in previous sections, when the planner considers the uncertainty-reduction goals, it does so for a particular plan instance. As a consequence of this, at the time of their evaluation, the uncertainty-reduction goals (which are expressed as symbolic algebraic inequalities) will be expressed in terms of specific bounded symbolic variables. Therefore, determining if an uncertainty-reduction goal is satisfied consists of a single evaluation (rather than a series of evaluations as was required in the geometric constraints). In particular, since the uncertainty-reduction goals are expressed as inequalities of the form $expr_1 < expr_2$, and since at least one of these expressions is always a single constant, if we find the maximum value for $expr_1$ and the minimum value for $expr_2$ (under the constraints contained in the world description), we can determine whether the uncertainty-reduction goals are met simply by checking to see if $\max(expr_1) < \min(expr_2)$.

In order to find upper and lower bounds on symbolic expressions, we have implemented a system similar to the SUP/INF system which was introduced by Bledsoe [2], and then refined by Shostak [30], and later Brooks for his ACRONYM system [4]. The functions SUP and INF each take two arguments, a symbolic expression and a set of variables, and return upper/lower bounds on the expression in terms of the variables in the variable set. The method SUP/INF employs is to recursively break down expressions into subexpressions, find bounds on these subexpressions, and then combine the bounds using rules from interval arithmetic. Obviously this works for linear expressions where superposition holds. When expressions are nonlinear, however, it is quite possible that the bounds on the individual subexpressions will be looser than the bounds on the subexpressions when considered in the context of the whole expression. Because of this, it is possible that SUP/INF will sometimes find bounds which are not exact.

In spite of this disadvantage, the policy of recursively finding bounds on subexpressions and then combining those bounds guarantees that the algorithms will terminate. This has been shown by Shostak for his version of SUP/INF, and later by Brooks for his modified versions. Furthermore, even though it is possible that SUP/INF will not return exact bounds, it has been shown (again, by Shostak and later by Brooks) that they are conservative, in that SUP always returns a value which is greater than or equal to the maximum, and INF always returns a value less than or equal to the minimum. The fact that SUP/INF sometime only approximates solutions is not a severe problem for SPAR, since failure to satisfy uncertainty constraints has a worst case result of the addition of sensing actions to the plan. That is, if the CMS determines that the uncertainty constraints cannot be satisfied, it does not backtrack. It merely prepares for the possibility of failure.

6. A Task Planning Example

In this section, we will illustrate SPAR's flow of control with an assembly example. Consider the assembly task shown in Figure 5. The assembly goal is to have the peg inserted into the block so that the small hole in the block is aligned with the hole in the peg's base. The user specifies this with a goal of the form:

assembled(peg,block,Msurfaces, Tm, Va)

where Msurfaces is instantiated to a two-element list, the first element being a list of the peg's surfaces which will come into contact with the block, and the second element being a list of the block's surfaces which will come into contact with the peg. The variable Tm is instantiated to a homogeneous transformation matrix which represents the goal position of the peg relative to the position of the block. The variable Va is instantiated to a vector which specifies the approach for the mating operation relative to the position of the block. In other words, the user specifies the positions of the parts relative to one another in the goal configuration, as well as the relative locations prior to the goal.

In order to satisfy this goal, SPAR examines its possible actions, and selects the assemble action. Of course, the assemble action has both operational and geometric preconditions which must now be considered, so the planner pushes these onto the appropriate goal stacks. The goal stacks and plan action list are shown in Figure 6.

At this point, a word about the meaning of the preconditions is in order. The assemble action has a precondition of the form:

```
geo(GoalId1,ActionId,
    in_position_class(Position,PositionList),
    part_location(Obj2,Position))
```

As we discussed in Section 5, SPAR associates a set of stable poses with each object. By stable pose, we mean an orientation in which the object will rest naturally on the table. In order to mate two objects, SPAR requires that the stationary object be in one of its stable poses which does not obscure any of its mating features. The set of stable poses which satisfy this condition is easily determined by comparing each stable pose's set of occluded faces with the set of mating features. When the planner adds the assemble action to the plan, it instantiates the variable PositionList to this list. (We should note that the list of stable poses is actually a list of pointers to the data structures for the stable poses.)

This same kind of instantiation takes place for the precondition

```
geo(GoalId2,ActionId,
    member(Grasp,GraspList),
    holding(Obj1,Grasp))
```

In our system, grasping configurations specify not only the geometric configuration which is used to grasp the object, but also the set of object features which are obscured by the grasp (as was discussed in Section 5.2.4). Therefore, it is a simple matter to determine which grasping configurations do not obscure the mating features of the object. When the planner adds the assemble action to the plan, it instantiates the variable GraspList to be this set of grasping configurations.

Figure 6 shows the instantiated versions of the preconditions for the assemble action as they appear on the goal stacks. Note that the variables used to identify the preconditions and the action to which the preconditions correspond have also been instantiated.

The first operational goal is that the gripper be holding the peg in some valid grasp (remember that at the operational level, SPAR is not concerned with the grasp beyond this condition). Since it is not possible to merely add a constraint to the plan to achieve this goal (i.e., there is no existing action in the plan whose execution can be constrained so that it results in the manipulator holding the peg), SPAR inserts the action pickup(peg,grasp_1) into the plan, with the constraint that the pickup action must occur prior to the mating action. This results in the addition of an arc to the ordering graph, directed from action2 to action1. The preconditions of the pickup action are then pushed onto the appropriate goal stacks. The resulting goal stacks are shown in Figure 7. Note that when the planner adds this action, it instantiates the variable Grasp to the label grasp_1, and that this instantiation affects all appearances of Grasp on the goal stacks.

The remaining operational goals are trivially satisfied by the initial world state, so the planner moves them to the satisfied goal list and turns to its geometric goals. (Note that when these goals are satisfied, instances of the variable Pos_1 and Pos_2 on the goal stack are instantiated to init_pos1 and init_pos2. The corresponding label sets are constrained to contain single elements which are the homogeneous transformations representing the initial positions of the block and peg.) The top goal on the geometric goal stack, goal_8, is for the

pickup action, and it specifies that the manipulator configuration used to pickup the peg, grasp_1, be physically realizable by the robot. To satisfy this goal, the planner attempts to add a constraint on the way in which grasp_1 is chosen, so that the configuration will be reachable. This is done by instructing the CMS to add the constraint reachable(grasp_1,init_pos2) to the constraint network. For our example, we will assume that this constraint is consistent with the constraint network.

The next goal on the geometric goal stack, goal_3, specifies that grasp_1 must not obscure any of the mating features of the peg. This is expressed as a member constraint, that is, a restriction on the label set for the plan variable grasp_1. Again, the planner invokes the CMS to add the member constraint to the constraint network. For the example, let us suppose that this succeeds. Note that if adding this constraint resulted in an inconsistent constraint set, SPAR would be forced to insert additional manipulations.

Up to this point in the example, SPAR has been able to satisfy geometric goals merely by adding constraints on the way in which operations are performed. In some cases, it will not be possible to satisfy geometric goals this way, and an alternative approach must be used. This is the case for the geometric precondition goal_4, which constrains the possible positions of the block. Consider the situation when the block is face down in the initial world state, as shown in Figure 5. Since there is no action currently in the plan which manipulates the block, SPAR cannot constrain the execution of a plan action to achieve the goal. Furthermore, the planner cannot add a constraint on the block's initial position, because it is a constant value which is defined by the initial world state. Therefore, backtracking must be used to find some alternative method to satisfy the goal goal_2, which specifies the position of the block.

Remember that goal_2 was originally satisfied by the initial world state. On backtracking, SPAR will try to find some other action in the plan to satisfy goal_2. As mentioned above, there is no action in the current plan which can accomplish this. Therefore, SPAR adds the action putdown(block,pos_1) to the plan. When SPAR reconsiders goal_4, the value of pos_1 will be constrained so that no mating features of the block are in contact with the table when the block is in this position. This amounts to constraining pos_1 to be any of the block's stable poses other than p2P3, the single configuration which obscures the hole. In addition, SPAR adds the constraint prior_to(action_3,action_1) to the ordering graph, since the block must be put down prior to the assemble action. Of course the addition of this plan action introduces new goals, and so additional planning must be done. This planning, however, is very similar to the planning which must be done to pick up the peg appropriately, and so we will not discuss it here.

The final result of the first phase of planning is shown in Figures 8–10. Figure 8 shows the four actions which are in the plan. The top of Figure 9 shows the geometric binary constraint network, which can be interpreted as follows. The grasping configuration grasp_2 is used to pick up the block, and then to place it on the table. Therefore, both init_pos_1 and pos_1 must be reachable using grasp_2. This is indicated by the arcs connecting grasp_2 to init_pos_1 and pos_1. Similarly, grasp_1 is used to pick up the peg, and then to assemble the peg to the block (which is now located in pos_1). The possible pairs of values for each of these

arcs are shown in Figure 10, as are the label sets for the nodes. The possible pairs of values for each arc in the network are determined by examining each possible pair of values from the label sets of the connected nodes and collecting those which meet the conditions outlined in Section 5.2.4. In the figure, we have represented stable poses by symbols of the form pxPy, where x is used to indicate the object (the peg is indicated by x=1, the block by x=2) and y is used to indicate the specific stable pose for the object. Symbols representing grasping configurations have a similar interpretation.

The bottom of Figure 9 shows the ordering graph. Note that in the ordering graph, in addition to the arcs we have mentioned above, there is an arc from action_3 to action_2. This arc is added to the graph because the pickup action used to pick up the block (action_4) clobbers the gripper(open) operational goal for the pickup action used to pick up the peg (action_2). To remedy this, the putdown action (action_3) is constrained to come between action_4 and action_2, to act as a white knight and reestablish the gripper(open) goal.

Once the operational and geometric goals have been satisfied, SPAR considers the uncertainty-reduction goals. As we have described earlier, SPAR chooses a specific instance of the plan (which satisfies the constraint network), and propagates uncertainties forward through the plan actions to determine if the uncertainty-reduction goals are satisfied. For this example, we will only consider the uncertainty-reduction goals for the first pickup action, which were discussed in Section 3.2.

In order to evaluate the constraints associated with these goals, SPAR invokes the procedure which constructs and evaluates the symbolic constraint for the goal. This involves symbolic matrix multiplication, symbolic matrix inversion, and symbolic algebraic simplification.

The resulting expression for the Y component of $P^{-1}C_1$:

$$-2.0 + 3.0*\cos(\text{thetagr})*\cos(\text{theta\_o}) + \cos(\text{thetagr})*\text{dx\_o} + \sin(\text{thetagr}) + \sin(\text{thetagr})*\text{dxgr} + -1.0*\cos(\text{thetagr})*\sin(\text{theta\_o}) + -1.5*\cos(\text{thetagr}) + -3.0*\sin(\text{thetagr})*\sin(\text{theta\_o}) + -1.0*\cos(\text{theta\_o})*\sin(\text{thetagr}) + -1.0*\sin(\text{thetagr})*\text{dy\_o} + -1*\cos(\text{thetagr})*\text{dygr}$$

Note that thetagr, dxgr, dygr and dzgr represent the uncertainties in the gripper configuration, and theta_o, dx_o, dy_o and dz_o represent the uncertainties in the object position. Also, for this particular plan instance, W_p was three inches and W_m was four inches. The complexity of this expression illustrates the reasons we outlined in Section 4.3 for applying uncertainty-reduction planning to specific plan instances instead of using a constraint posting approach.

Similar expressions are found for the remaining terms, but we will omit these here. Using the SUP and INF routines given the bounds on the uncertainties listed in Table 2, the lower bound on this expression is found to be 3.2793, which indicates that the constraint was satisfied. The remaining three constraints are evaluated in a similar fashion.

If the uncertainty-reduction goals are not satisfied in the world description, SPAR attempts to add a sensing operation to the plan. Since it is impossible to predict the results of a sensing action, the add/delete lists for sensing actions merely describe the uncertainty in the object's location after the application of the sensing operation. If this reduction is sufficient, the sensing operation is inserted into the plan.

If SPAR cannot sufficiently reduce the uncertainty in the peg's location, it augments the plan instance with verification-sensing operations and local recovery plans.

7. Conclusions

This paper represents a step toward a planning system which can create assembly plans given as input a high level description of assembly goals, geometric models of the components of the assembly, and a description of the capabilities of the work cell (including the robot and the sensory system). The resulting planner, SPAR, reasons at three levels of abstraction: the operational level (where high-level operations are planned), the geometric level (where geometric configurations of the actions are planned) and the uncertainty-reduction level (where world uncertainties are taken into account).

At the first two levels of planning, we have extended the constraint posting approach to domain-independent planning by adding geometric preconditions to the actions, linking these to operational goals via plan variables, and expanding the CMS to be able to deal with geometric constraints. At the uncertainty-reduction level of planning, we have expressed uncertainties in the world in terms of homogeneous transformations whose elements are defined in terms of symbolic uncertainty variables. We then expressed limits on tolerable uncertainties in terms of operations on transformations. When the uncertainty-reduction goals cannot be satisfied, rather than abandon the plan, our system augments the plan with sensing operations for verification, and when possible, with local error recovery plans.

At this point, there are a number of areas in our system which are either ad hoc, or require far too much input from the user. For example, the local error recovery plans must be entered by the user, and associated with the uncertainty-reduction goals a priori. One goal of our future work will be to automate this process by employing geometric reasoning about possible errors and error recovery. A further shortcoming of SPAR is the lack of any sort of motion planning system. Incorporating a motion planner with the current system is another goal of our future work.

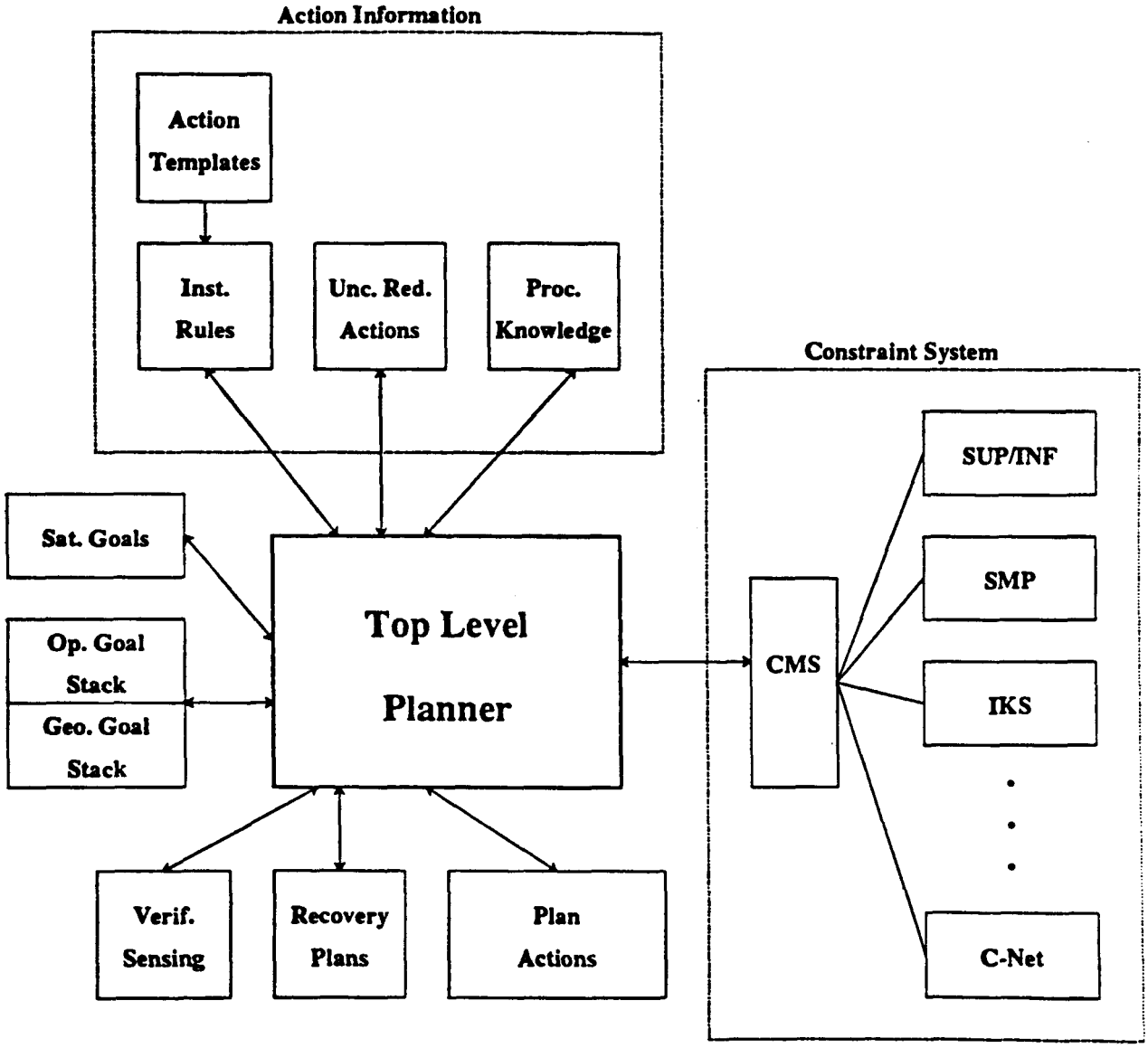


Figure 1. Block Diagram of SPAR

action-id: ActionId,

action: pickup(Object,Grasp),

preconditions:

operational:

op(G1,ActionId, gripper(open))
op(G2,ActionId, part_location(Object,Pos))

geometric:

geo(G3,ActionId,
reachable(Grasp,Pos),
part_location(Object,Pos))

uncertainty-reduction:

$0 < [0,1,0,0]\, P_1^{-1}\, C_1$
$0 > [0,1,0,0]\, P_1^{-1}\, C_2$
$0 < [0,1,0,0]\, P_2^{-1}\, C_1$
$0 > [0,1,0,0]\, P_2^{-1}\, C_2$

add-list:

holding(Object,Grasp)
part_location_unc(Object,NewUnc)
gripper(closed)

delete-list:

part_location(Object,Pos)
part_location_unc(Object,OldUnc)
gripper(open)

Figure 2. The Action Template for the Pickup Action

If the Goal is "holding(Object,Grasp)" then:

Generate a unique symbol for Grasp

Set GraspList to the list of grasping configurations for Object

Set Initial Constraint List to contain "member(Grasp,GraspList)"

Generate unique symbols for: ActionId, G1, G2, G3

Set Template =

```
action( ActionId,
        pickup(Object,Grasp),
        preconditions(
                [op(G1,ActionId, gripper(open)),
                 op(G2,ActionId, part_location(Object,Pos))],

                [geo(G3,ActionId,
                        reachable(Grasp,Pos),
                        part_location(Object,Pos))]),

        addlist([holding(Object,Grasp),
                 gripper(closed)]),

        dellist([part_location(Object,Pos),
                 gripper(open)]))
```

Figure 3. Rule to Instantiate an Action Template

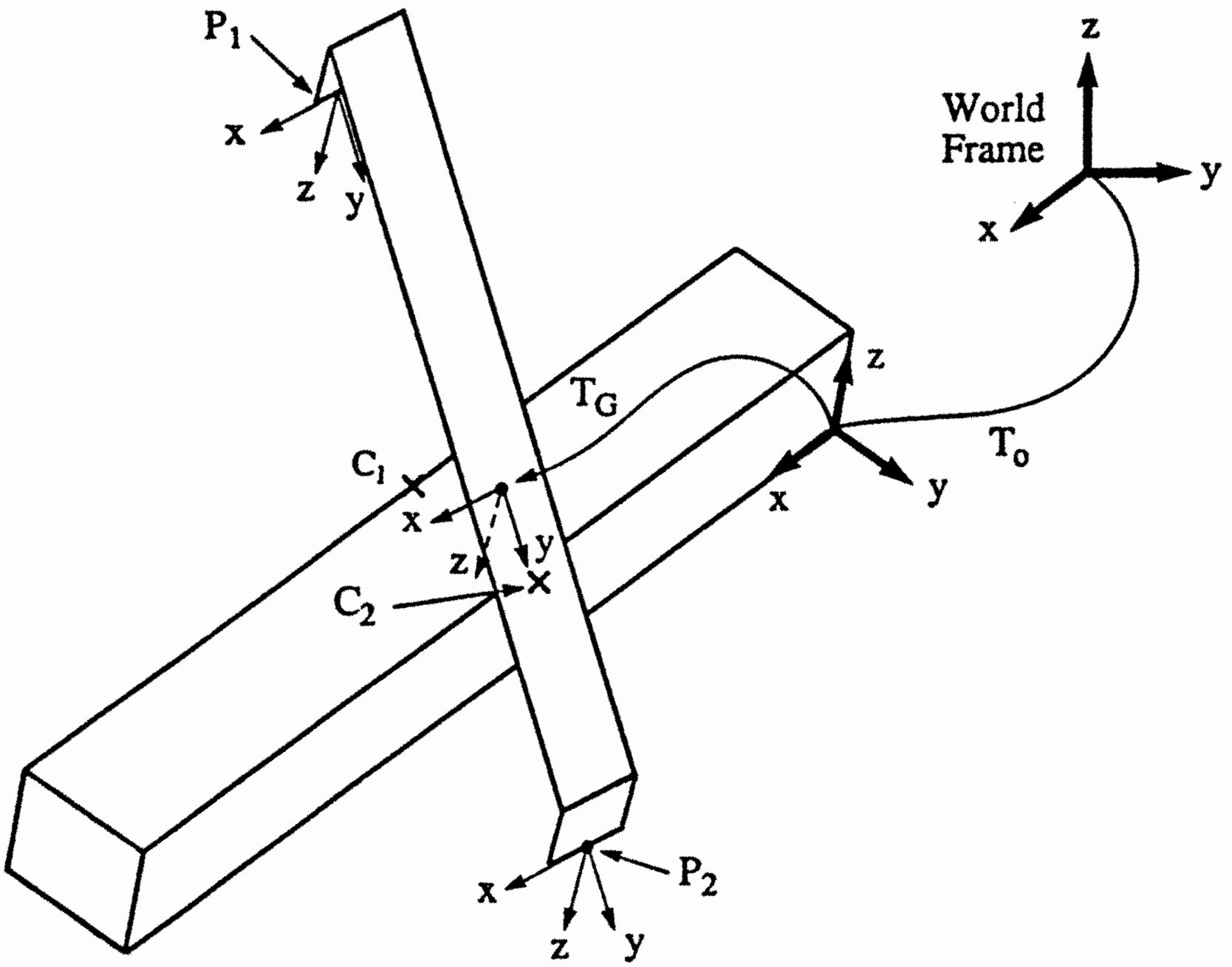


Figure 4. Possible Finger Coordinate Frames and Contact Points for the Pickup Action.

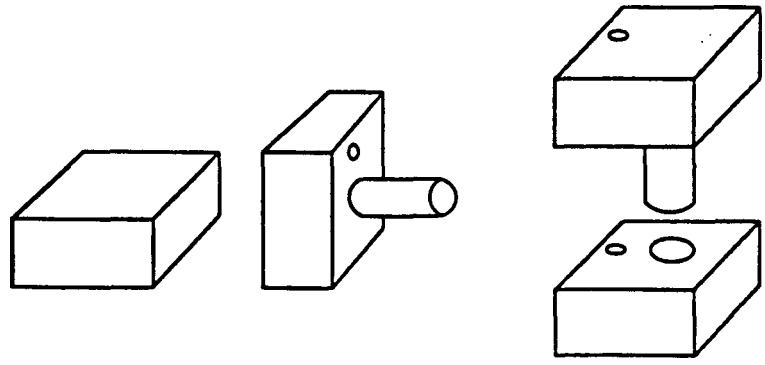


Figure 5. The Initial State and Assembly Goal for the Example

OPERATIONAL GOAL STACK

op(goal_1, action_1, holding(peg, Grasp))

op(goal_2, action_1, part_location(block, Pos_1))

GEOMETRIC GOAL STACK

geo(goal_3, action_1,
member(Grasp, [p1G1, ... p1Gn]),
holding(peg, Grasp))

geo(goal_4, action_1,
in_position_class(Pos_1, [p2P1, p2P2, ... p2P6]),
part_location(block, Pos_1))

geo(goal_5, action_1,
mate_reachable(Grasp, Pos_1, trans_1),
part_location(block, Pos_1)
holding(peg, Grasp))

PLAN ACTIONS

action(action_1,assemble(
peg,
block,
[[peg_face], [block_face]],
trans_1, vec_1))

Figure 6. Goal Stacks and Plan Actions after the Addition of the "Assemble" Action

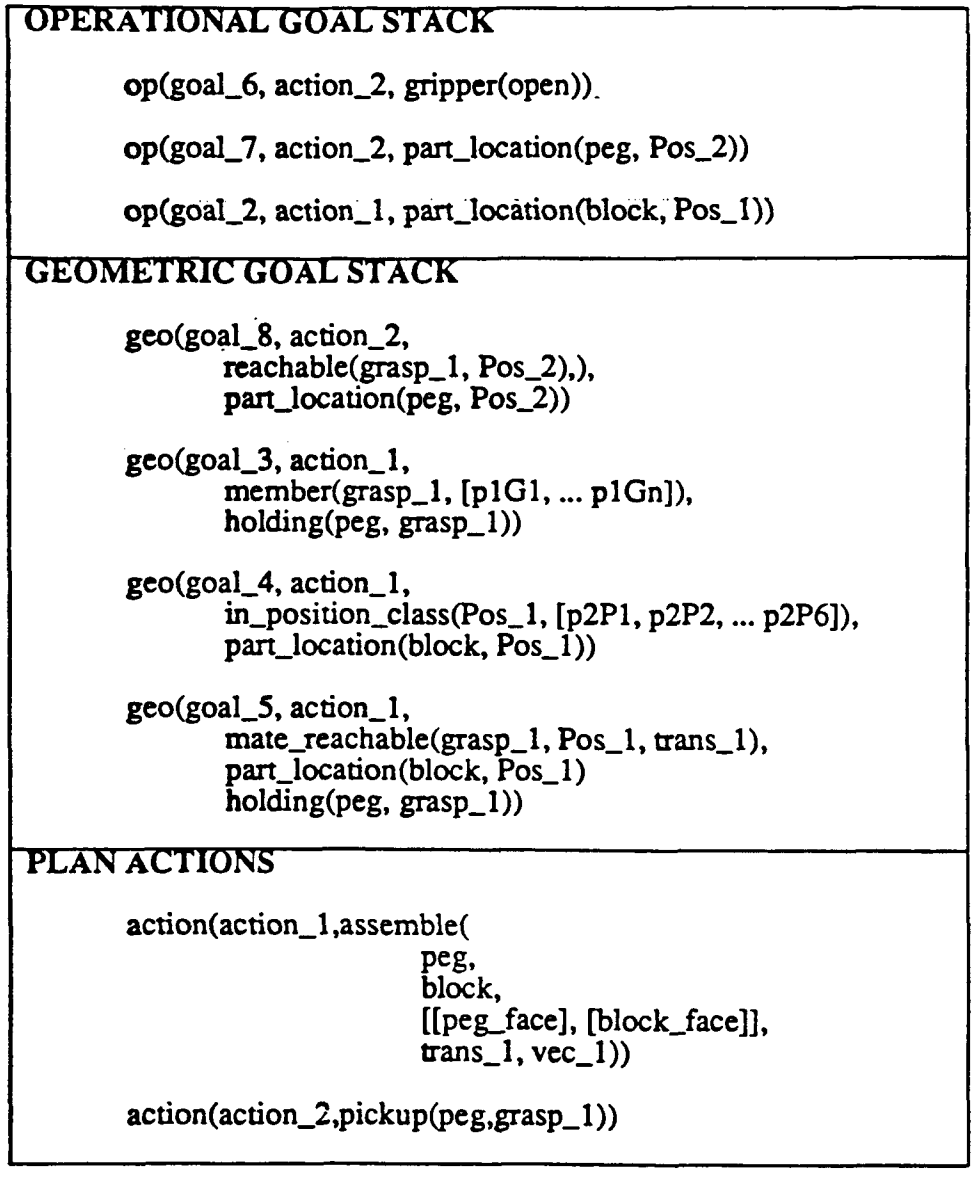


Figure 7. Goal Stacks and Plan Actions after the Addition of the "Pickup" Action

Plan Actions

action(action_4, pickup(block, grasp_2))

action(action_3, putdown(block, pos_1))

action(action_2, pickup(peg, grasp_1))

action(action_1, assemble(
peg,
block,
[[peg_face], [block_face]],
trans_1,
vec_1))

Figure 8. Plan Actions Solve the Assembly Task Shown in Figure 5

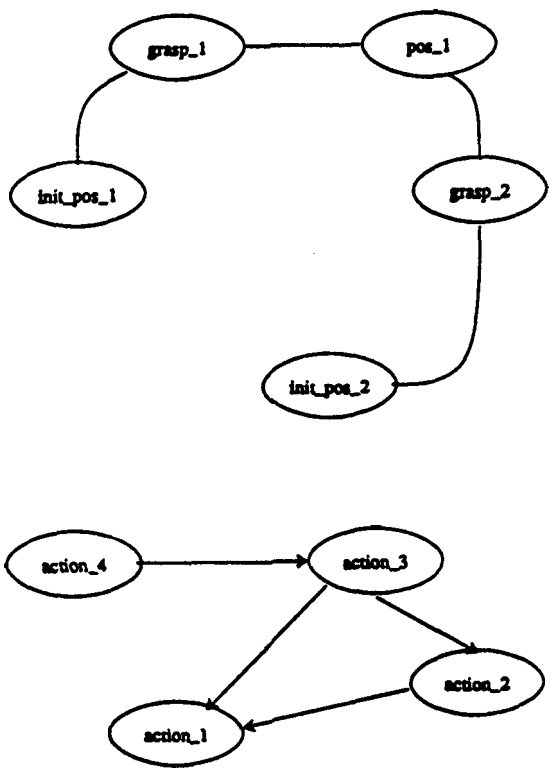


Figure 9. Constraint Network for the Assembly Plan to Solve the Task Shown in Figure 5

NETWORK ARCS

grasp_1-pos_1:

[p1G1/p2P1,p1G1/p2P2,p1G1/p2P4,p1G1/p2P5,p1G1/p2P6,p1G11/p2P1,
p1G11/p2P2,p1G11/p2P4,p1G11/p2P5,p1G11/p2P6,p1G12/p2P1,p1G12/p2P2,
p1G12/p2P4,p1G12/p2P5,p1G12/p2P6,p1G13/p2P1,p1G13/p2P2,p1G13/p2P4,
p1G13/p2P5,p1G13/p2P6,p1G14/p2P1,p1G14/p2P2,p1G14/p2P4,p1G14/p2P5,
p1G14/p2P6,p1G15/p2P1,p1G15/p2P2,p1G15/p2P4,p1G15/p2P5,p1G15/p2P6,
p1G18/p2P4,p1G18/p2P5,p1G18/p2P6,p1G20/p2P1,p1G20/p2P2,p1G20/p2P4,
p1G20/p2P5,p1G20/p2P6,p1G3/p2P1,p1G3/p2P2,p1G3/p2P4,p1G3/p2P5,
p1G3/p2P6,p1G6/p2P1,p1G6/p2P2,p1G6/p2P4,p1G6/p2P5,p1G6/p2P6,p1G7/p2P1,
p1G7/p2P2,p1G7/p2P4,p1G7/p2P5,p1G7/p2P6,p1G8/p2P1,p1G8/p2P2,p1G8/p2P4,
p1G8/p2P5,p1G8/p2P6,p1G9/p2P1,p1G9/p2P2,p1G9/p2P4,p1G9/p2P5,p1G9/p2P6]

grasp_2-pos_1:

[p2G1/p2P1,p2G1/p2P4,p2G10/p2P2,p2G10/p2P4,p2G11/p2P2,p2G11/p2P6,
p2G12/p2P2,p2G12/p2P5,p2G13/p2P1,p2G13/p2P5,p2G13/p2P6,p2G14/p2P1,
p2G14/p2P6,p2G15/p2P1,p2G15/p2P5,p2G16/p2P5,p2G2/p2P1,p2G3/p2P4,
p2G4/p2P5,p2G4/p2P6,p2G5/p2P5,p2G5/p2P6,p2G6/p2P5,p2G6/p2P6,
p2G7/p2P6,p2G8/p2P1,p2G8/p2P2,p2G8/p2P4,p2G9/p2P1,p2G9/p2P2])

grasp_2-init_pos2:

[p2G1/tr2,p2G16/tr2,p2G2/tr2,p2G3/tr2,p2G4/tr2,p2G5/tr2,p2G6/tr2,p2G7/tr2]

grasp_1-init_pos1:

[p1G12/tr1,p1G3/tr1,p1G7/tr1,p1G8/tr1]

LABEL SETS

pos_1: [p2P1,p2P2,p2P4,p2P5,p2P6]

init_pos1: [tr1]

init_pos2: [tr2]

grasp_2: [p2G1,p2G2,p2G3,p2G4,p2G5,p2G6,p2G7,p2G8,p2G9,
p2G10,p2G11,p2G12,p2G13,p2G14,p2G15,p2G16]

grasp_1: [p1G1,p1G3,p1G6,p1G7,p1G8,p1G9,p1G11,p1G12,p1G13,
p1G14,p1G15,p1G17,p1G18,p1G20]

Figure 10. The Arcs and Label Sets for the Constraint Network Shown in Figure 9

Table 1. The Constraints which Are Used in the First Phase of Planning

prior_to(Action1,Action2):

Action1 must be executed prior to Action2.

reachable(Grasp,Position):

Grasp is a grasping configuration which must be physically realizable when grasping an object located in Position.

mate_reachable(Grasp,P,T):

P defines a fixed coordinate frame. T defines the destination coordinate frame of the object which is held in the manipulator, relative to P. Grasp is a grasping configuration which must be physically realizable given T and P.

member(Item,List):

Item must be chosen from List.

in_position_class(Position,PositionList):

Position must be an element of PositionList.

Table 2. Bounds on Uncertainty Variables Used in the Example of Section 6

Bounds on Uncertainty Variables		
Variable	Lower Bound	Upper Bound
dxgr	-0.001	0.001
dygr	-0.001	0.001
dzgr	-0.001	0.001
thetagr	-0.001	0.001
dx_o	-0.11	0.11
dy_o	-0.11	0.11
dz_o	-0.11	0.11
theta_o	-5.5	5.5

References

[1] Amber, A. P., and Popplestone, R. J., Inferring the Positions of Bodies from Specified Spatial Relationships. *Artificial Intelligence*, 6:157–174.

[2] Bledsoe, W. W, The SUP-INF method in Presburger Arithmetic. U. of Texas at Austin Math. Dept. Memo ATP-18, December 1974.

[3] Boyer, M., and Daneshmend, L. K., An Expert System for Robot Error Recovery Computer Vision and Robotics Laboratory of McGill University, Technical Report TR-CIM-87-18, October 1987.

[4] Brooks, R. A., Symbolic Reasoning Among 3D Models and 2D Images. *Artificial Intelligence,* 17:285–348 1981.

[5] Brooks, R. A., Symbolic Error Analysis and Robot Planning. *The International Journal of Robotics Research*, 1 (4), Winter 1982.

[6] Brost, R. C., Planning Robot Grasping Motions in the Presence of Uncertainty. Computer Science Department of Carnegie-Mellon University Technical Report CMU-RI-TR-85-12, July 1985.

[7] Chapman, D., Planning for Conjuctive Goals. *Artificial Intelligence,* 32 (3):333-378, July 1987.

[8] Davis, E., Constraint Propagation with Interval Labels. *Artificial Intelligence*, 32 (3):281–331, July 1987.

[9] Dechter, R., and Pearl, J., Network-Based Heuristics for Constraint-Satisfaction Problems. *Artificial Intelligence*, 34 (1):1–38, December 1987.

[10] Donald, B. R., Robot Motion Planning with Uncertainty in the Geometric Models of the Robot and Environment: A Formal Framework for Error Detection and Recovery. *Proc. of the IEEE Int'l Conf. on Robotics and Automation,* 1588–1593 1986.

[11] Donald, B. R, A Search Algorithm for Motion Planning with Six Degrees of Freedom. *Artificial Intelligence*, 31 (3):295–353 March 1987.

[12] Durrant-Whyte, H. F., Uncertain Geometry in Robotics. *The IEEE Journal of Robotics and Automation,* 4 (1):23–31 February 1988.

[13] Erdmann, M. A., On Motion Planning with Uncertainty. MIT AI Lab Technical Report AI-TR-810, 1984.

[14] Fikes, R. E., and Nilsson, N. J., STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189–208, 1971.

[15] Gini, M. Doshi, R. Gluch, M., Smith, R. and Zualkernan, I., The Role of Knowledge in the Architecture of a Robust Robot Control. *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 561–567, 1985.

[16] Gottschlich, S. N., and Kak, A. C., A Dynamic Approach to High Precision Parts Mating. *Proc. of the IEEE Int'l Conf. on Robotics and Automation,* pp. 1246–1253, 1988. [An update version of this paper will appear in a forthcoming issue of *IEEE Trans. on Systems, Man and Cybernetics*..]

[17] Hutchinson, S. A., and Kak, A. C., A Task Planner for Simultaneous Fulfillment of Operational, Geometric and Uncertainty-Reduction Goals. Purdue University Technical Report, TR-EE 88–46, September 1988.

[18] Hutchinson, S. A., and Kak, A. C., Applying Uncertain Reasoning to Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities," *Proc. of the IEEE Symposium on Intelligent Control*, 1988.

[19] Hutchinson, S. A., Cromwell, R. L., and Kak, A. C., Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities. *Proc. of the IEEE Int'l Conf. on Robotics and Automation,* 1068–1075, 1988.

[20] Lee, C. S. G,. and Ziegler, M., A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots. *Proc. of the Thirteenth Int'l Symposium on Ind. Robotics and Robots 7*, Chicago IL, April 1983.

[21] Lozano-Perez, T., A Simple Motion-Planning Algorithm for General Robot Manipulators. *The IEEE Journal of Robotics and Automation,* RA-3:224–239, No. 3, June 1987.

[22] Lozano-Perez, T., and Brooks, R. A., An Approach to Automatic Robot Programming. MIT AI Lab, AIM 842, 1985.

[23] Lozano-Perez, T., Jones, J. L., Mazer, E., O'Donnell, P. A, Grimson, W. E. L., Tournassound, P., and Lanusse, A., Handey: A Robot System that Recognizes, Plans and Manipulates. *Proc. of the IEEE Int'l Conf. on Robotics and Automation,* 1987.

[24] Lozano-Perez, T., Mason, M. T., and Taylor, R. H., Automatic Synthesis of Fine-Motion Strategies for Robots. *The Int'l Journal of Robotics Research,* 3 (1) Spring 1984.

[25] Nof, S. Y., Maimon, O. Z., and Wilhelm, R. G., Experiments for Planning Error-Recovery Programs in Robotic Work. Purdue Univ. School of Industrial Engineering Resezrch Memo No. 87-2, March 1987.

[26] Pertin-Troccza, J., On-Line Automatic Robot Programming: A Case Study in Grasping. *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1987.

[27] Pertin-Troccaz, J., and Puget, P., Dealing with Uncertainties in Robot Planning Using Program Proving Techniques," *Proc. of the Fourth Int'l Symposium of Robotic Research,* Aug. 1987.

[28] Popplestone, R. J., Ambler, A. P., and Bellos, I. M., A Language for Describing Assemblies. *The Industrial Robot*, 131–137 September 1978.

[29] Sacerdoti, E. D., *A Structure for Plans and Behavior,* Elsevier North-Holland, Inc., New York, 1977.

[30] Shostak, R. E., On the SUP-INF Method for Proving Presburger Formulas. *Journal of the ACM*, 24 (4):529–543, October 1977.

[31] Smith, R. C., Cheeseman, P., On the Representation and Estimation of Spatial Uncertainty. *The Int'l Journal of Robotics Research*, 5 (4), Winter 1986.

[32] Smith, R. E., and Gini, M., Reliable Real-time Robot Operation Employing Intelligent Forward Recovery," *Journal of Robotic Systems,* 3 (3):281–300, 1986.

[33] Srinivias, S., Error Recovery in Robots Through Failure Reason Analysis. *Proc. of the AFIPS National Computer Conference,* 275–282 1978.

[34] Stefik, M., "Planning with Constraints (MOLGEN: Part 1). *Artificial Intelligence,* 16:111–140 1981.

[35] Tournassound, P., and Lozano-Perez, T., Regrasping. *Proc. of the IEEE Int'l Conf. on Robotics and Automation,* 1924–1928 1987.

[36] Wilkins, D. E., Representation in a Domain-Independent Planner. *Proc. Eigth Int'l Joint Conf. Artificial Intelligence,* 733–740, 1983.

Blank Page

Dynamic Task Allocation and Execution Monitoring in Teams of Cooperating Humans and Robots

S. Y. Harmon

Robot Intelligence International
San Diego, CA 92107-7890

Abstract

Dynamic task allocation and execution monitoring are critical elements for the effectiveness of teams of cooperating humans and robots. Execution monitoring is necessary to identify when and where dynamic allocation is needed. Dynamic task allocation enables the resources of the team to be efficiently mapped onto the demands of the prevailing task regardless of its changing nature and uncertainties. The technology to realize dynamic task allocation comes from the areas of distributed problem solving, multiple coordinated robots, error recovery, scheduling and human–machine interactions.

Keywords: dynamic task allocation, execution monitoring, cooperating robots, distributed artificial intelligence.

Introduction

The benefits of coordinated teams of intelligent individuals whether humans or robots or combinations of the two have been recognized for their considerable potential. Foremost in real applications, coordinated teams can have increased reliability amidst failures and attrition through decreased sensitivity to individual failures and improved graceful system degradation.[1–5] In addition, the performance of a team can be more accurate than individuals when faced with task uncertainty and incompleteness. If properly designed, teams can provide increased spatial coverage and improved geometric advantage and can cope with the communications and processing bandwidth requirements associated with large numbers of sensors and effectors which are required to realize these advantages.[1,2,5] The efficiency of a team can be significantly better than uncoordinated individuals through parallelism, specialization and resource sharing.[1] In fact, the success of scientific research has been attributed to its parallelism and pluralism.[6] The modular nature inherent to teams make them more adaptable to change through task matching and extensibility.[1–5] Finally, for complex tasks, teams offer more cost effective control of system complexity through reduced hardware and software complexity,[1-5] enhanced real time response[2,3] and lower communications and processing costs.[2]

Execution monitoring and dynamic task allocation in teams of cooperating humans and robots draw support from a diversity of supporting technologies including distributed problem solving, multiple cooperating robots, scheduling, error recovery and human–machine

interactions. The influences of these technologies are discussed below. The use of teams brings with it the need to consider the issues of distributed system design. Due to space limitations, only a few of the critical design issues of distributed systems are discussed below and the reader is referred to the literature which is cited for further important information. Execution monitoring is discussed first since it introduces the need for dynamic task allocation. Then, the issues of dynamic task allocation are discussed. Finally, a few conclusions to the discussion are presented, and a number of remaining research issues are identified.

Supporting Technologies

Researchers in distributed artificial intelligence (DAI) have been examining issues related to interacting intelligent entities for many years.[4–7] A particularly relevant subset of DAI is distributed problem solving (DPS). The coordination problem of intelligent entities is a subset of DPS.[8]

Interest in developing systems of cooperating robots has grown significantly over the past few years. Considerable work has been devoted to coordinating multiple robots simply to prevent collisions while working in common geometric spaces.[9–15] Some work has been devoted to developing systems in which robots actually cooperate to achieve common goals. Applications of this work include multiple robots in a single work cell,[16] cooperating underwater robots,[17] and multiple cooperating work stations.[18–19]

An enormous amount of work has been done in the area of human–machine interaction. Extensive review of this literature is well outside the scope of this paper; however, a small subset of this work has addressed human interaction with automated resources which directly relates to the problems of task allocation and execution monitoring in human–robot teams. Applications for this work include such areas as human–autopilot interactions,[20] multioperator spacecraft tracking,[21] computer-human teams for ill–structured (i.e., not well understood) risk management problems,[22] and supervisory control of automated systems for flexible manufacturing systems,[23]

Scheduling and resource allocation have received considerable interest in the robotics community for coordinating the actions of multiple work cells. Extensive review of this vast area is also outside the scope of this paper, but a few observations are appropriate.

Scheduling and resource allocation have been applied to such applications as scheduling multiple assembly robots[24–33] and routing automated guided vehicles.[34,35] Several scheduling and resource allocation techniques have been explored for automation of various types, including dynamic programming and queuing theory,[29] control theory and optimization,[30] essentially decision free Petri nets for resource allocation,[32] search techniques,[28,34] a combination of graph theory, optimization and A* search,[35] nonlinear planning techniques,[35] and various opportunistic (i.e., run time) scheduling techniques.[24,25,27,28,31,33] Opportunistic scheduling is most directly related to the problem of dynamic task allocation in human–robot teams since it deals directly with the assignment of

resources to tasks (or vice versa) after the task has begun.

Much of the work on error detection and recovery has implications for execution monitoring and dynamic allocation. Error detection techniques have been developed to find faults in computer systems and to generate discriminatory tests,[36,37] to recover from errors in single robots for assembly[38–41] and reactor maintenance,[42] to recover from errors encountered when executing mobile robot route plans,[43] and to detect and diagnose plan faults.[44] Most of these techniques account only for errors caused by uncertainties in accurate and correct task descriptions.[40] Although almost no formal analysis of the error recovery problem is available, a limited theory for error detection and recovery using configuration space representation and a forward chaining planner has been introduced.[45]

Distributed Cooperating Systems Design

The capabilities of automated systems which can be distributed include sensing and interpretation, planning and problem solving, controls and actuators, and knowledge and computing. Several questions arise when considering designs which distribute these capabilities. The distribution of capabilities is relevant to any kind of distributed system whether manned or automated; however, combining humans with automation presents a number of human–machine design options which are not available with strictly automated systems. The humans and machines can adopt any of several possible roles including teleoperation, autonomous entities, or mixtures of the two.

The organization of a team of distributed humans and robots is the plan by which resources are initially distributed and exchanged throughout the task. The choice of an organization is perhaps the second most important decision a system architect must make. The first most important decision is the choice of the task. An organization can be characterized by the structures of its communication, task, power, and skills and by the distribution of its knowledge, resources, and perception. An organization has been described as the system of constraints on the individual entity.[7]

There are essentially two distinct types of organizations, hierarchies and committees. Each of these has different attributes which are appropriate for different situations. In general, the amount of cooperation within a distributed system is predetermined by the problem domain. Committees are data driven and hierarchies are goal- driven.[4] Committees tend to maximize system flexibility through communication, whereas hierarchies tend to minimize communication through structure.

A system organization can be created at design time and remain static throughout the execution of the task, or it can be dynamic. A dynamic organization can be created once at the beginning of the task and remain static thereafter, or it can be modified during task execution. A dynamic organization may be more costly to implement but it permits reorganization in the event of failures. In addition, dynamic organizations are generally needed in problems where little is known a priori.[4] Further, as the number of agents in a team

becomes very large then a dynamic organization may be necessary.[7]

Several paradigms for dynamic organization control have been proposed.[4] For instance, one technique uses epsilon-optimal learning automata to heuristically select the hierarchical structures to be formed. This approach improved the speed of convergence onto a solution but the optimum resultant organizations have yet to be found.[46] Negotiation systems such as the contract net formalism provide another alternative for controlling system reorganization.[47] However, negotiation systems may not be sufficiently responsive to external events and may require a significant amount of communications bandwidth for the negotiation process. Audience restriction reduces this bandwidth requirement and is particularly favorable when agent addition or deletion decreases and specialization increases.[4] Regarding the distribution of sensing and actuation resources as well as others, heterogeneous agents seem to complicate design whereas homogeneous agents tend to simplify analysis.[7]

An important problem which is inherent to planning in distributed systems is maintaining coherence within the system. Coherence refers to the consistency of local plans and actions of the team members with the global plan of the entire team. Needless to say, global coherence is critical when using multiple problem solvers.[7] Partial coherence wastes resources and degrades performance.[48] One solution to the coherence problem is to guide each agent by a high level strategic plan for cooperation, taking care not to constrain each individual too much.[48] Another potential solution is to provide each agent with the ability to reason about its current state of problem solving and to predict the future actions of companion agents. This can be done by having each agent compute an abstracted high level description of its local state and from this description to formulate new high level goals and generate plans to achieve them. These high level goals and plans are then communicated to the other active agents.[48]

There are several reasons for distributing control. Centralized coordination seriously hampers system reliability, and mutual agreement contracts before action are inefficient and may not lead to acceptable solutions.[48] However, the distribution of control is complicated by the difficulty in achieving globally coherent behavior because each node lacks a global perspective of the state of the problem. Furthermore, conflicts must be avoided to realize the improved performance which is promised by distributed systems.[47] In addition, with increased coordination and cooperation comes increased communication.[4] Fortunately, task and coordination complexity can be reduced by minimizing unnecessary interaction between units.[47]

Execution Monitoring

The information for dynamic allocation comes from monitoring the execution of the task. Execution monitoring compares the information received through sensors and communications with that stored in models of expectations, behaviors, physical processes, sensors, actuators, and plans to detect situations where replanning and reallocation are necessary. Execution monitoring identifies inefficient utilization of team resources due to poor plans or to a task environment which no longer meets the criteria for plan or action validity. Execution

monitoring also identifies the occurrence of errors and failures within the components of the team.

All execution monitoring should be preplanned. If a priori expectations of execution results are not available, then effective execution monitoring is simply not possible. However, generating expectations is not enough to guarantee effective execution monitoring. Many other conditions must be met. The important components of the execution process must be sensed. No monitoring is possible if no sensors are available which provide information on the results of execution. Execution monitoring is extremely useful if the execution process is nondeterministic. If the process is well characterized and no deviations from this characterization are anticipated, then monitoring is probably not necessary. On the other hand, if little can be accurately predicted about the outcome of execution, then the process must be closely monitored so remedial actions can be taken as soon as possible. The particular execution process must be important to the success of accomplishing the task. If the process is not very important, then do not monitor it. If it is very important, then monitor it closely. Finally, some corrective action must be within the system's capabilities if the execution process begins to go awry. In other words, monitoring is only useful if something can be done about the problem once it is sensed. The more actions which are available to the system the more useful detailed monitoring is to decide which actions are most useful.

The factors which contribute to uncertainty in execution (and, therefore, to enhance the utility of execution monitoring) include uncertainties in the environment (how well is the environment understood), uncertainties in the accuracy of the plan (how well is the planning process for a particular task understood), and uncertainties in the system (how well are the system's behavior and the outcome of its actions understood). Execution monitoring provides a means of dealing with the effects of these uncertainties. No more execution monitoring should be preplanned or incorporated in the system than is useful.

Some execution monitoring issues include error detection, out of limits situations, questions of allocation/reallocation, and when to abort. In addition, one must ask what agents can be trusted and who can monitor what tasks. In tracking tasks, the progress made in closing toward the end goal and the violation of task constraints must be monitored. The monitoring task must consider task allocation, capability overlap, allocation overlap, sensing performance, and the models used to monitor execution.

Execution monitoring involves one or more internal models, mechanisms for detecting unacceptable situations, and procedures for recovery. These elements are usually organized into some structure of components to perform planning, monitoring, and recovery.

Execution Monitoring Structures

The structures of execution monitoring systems generally include a planner to generate a plan,[43,49] a monitor to identify deviations from the plan,[39,43,49,50] and a recoverer to devise a recovery strategy.[39] The monitor must cooperate with either the recoverer or the planner to develop a successful recovery plan. The planner must supply error, transitions to the monitor to help recognize errors and the monitor in turn helps the planner by retracing

steps and exploring to overcome errors.[43] Likewise, the recoverer uses the monitor to run further tests for more information.[39] Execution monitoring can reduce errors and improve performance of a DPS system through meta-level control which detects and diagnoses plan faults.[44]

Internal Models

Several different internal models are possible including models of expected behavior,[36,40,43,45,50,51] system structure,[35] sensors (e.g., limits of calibration), the task,[40] expected performance [44] and interactions between system components. Modeling techniques which have been used include finite state transition graphs with nodes of possible internal states and links representing transitions from one state to another,[51] physical models,[38,40,45] Petri nets,[37] finite state automata to map events to actions,[40] common sense heuristics representing useful facts,[40] and scripts developed from experiences.[43] Various reasoning mechanisms have been used with these models including mathematical techniques,[45] finite state automata,[40,51] deductive inference,[36,37,40,50] common sense reasoning,[38] plan simulation,[50] and formal logic.[51] Of the possible modelling techniques for describing behavior, simple rules are useful if behavior is simple. Petri nets are useful if the focus is on parallel events and unrestricted code is the last resort if all else fails.[37] Humans can acquire these models through training and robots can acquire these models through programming and/or training. Much of the representation must be the same for both planner and execution monitor to make communications and sharing of world models possible.[49]

Modelling uncertainties and accounting for errors help, but the real world is so complex that extremely sophisticated models may be required.[38] Often the size and complexity of the model and the associated data base require some focussing mechanism to make real time execution monitoring possible. Focussing mechanisms which have been implemented for DPS and robot error recovery include hierarchical organization of the data[36,37] and domain specific heuristics to present only the most relevant features of sensor data to the execution monitor.[43]

Detection Mechanisms

Detection mechanisms identify when something is going wrong during task execution and what it is. This could involve exceeding the capabilities of sensors, actuators, knowledge, or computing, or it could involve deviating from plan expectations. Some error detection mechanisms include sensor monitoring, hard-wired interrupts, and operator supervision.[42] Monitoring involves looking at such quantities as robot motion,[40] the motion of task objects,[40] raw sensor data,[41,42] high level sensor data representations,[50] and internal data base consistency.[44] Sensor data can be compared with performance thresholds,[42] data recorded from previous similar actions,[43] and the expected outcomes of a plan.[39,44,49] Once a deviation is detected the system can be interrupted,[41] an operator called,[42] or a plan recovery mechanism invoked.[39,40,43,49] Often in monitoring performance it is especially important to maintain pace with the real time evolution of events. Some measures to improve monitoring performance include using hard-wired interrupts to detect critical subsystem fail-

ures,[42] using a special sensor handler to filter data for only unexpected events,[39] using a dedicated processor to monitor sensor data[41] and precomputing critical raw sensor data thresholds to minimize computing associated with the monitoring process.[41] Representations of sensor data have varied from raw sensor data values[41] to high level predicates.[50]

An interesting issue related to sensor data monitoring is how to determine where the error is occurring. Suppose an error has been identified. Is the error caused by deviation from the plan, by the sensor producing the reading, by a faulty set of expectations, or by an actuator which did the wrong thing? A classical method for identifying faulty elements in a system is voting, but that works only when there is sufficient overlap in functionality which is often not the case in robot systems. If humans are involved, should the human trust the robot or vice versa? The most intuitive answer is that the human should be the final authority, but the recent incident aboard the USS Vincennes showed that multiple responsible humans were at fault and that the automated assistance was correct.

Recovery

There are several methods of handling errors once they are discovered. Task execution can proceed regardless of the error, but this is dangerous. Execution can be stopped when an error is detected, but this is inefficient. Preprogrammed error checking and recovery for every error situation is possible, but it is unreliable and expensive to design and to compute. A program can be generated from the task description which is guaranteed to execute correctly even in the presence of errors and uncertainties but this requires models of robot kinematics and dynamics and of such complex physical properties as friction and is generally successful only for fine motions. Finally, it is possible to analyze the error situation and replan but this is as yet an underdeveloped option.[38]

Most of the systems reviewed halted execution when an error was discovered.[40,42–44]) Then control was passed either to a human operator[42] or to a recovery module which diagnosed the problem and created a recovery plan.[40,43,44] At this point, discrepancies can be analyzed to determine whether error (1) is unimportant, (2) has a fixed solution or (3) requires complete replanning.[49] The repair could be simple (e.g., wait for mechanical wobble to stabilize) or complex (requiring complete revision of program). Revision of the complete program is a difficult job which generally involves considerable reasoning from a global knowledge base.[39] Several approaches attempted to patch the existing plan.[18,40,43.44]

Different techniques which have been suggested for plan recovery include a rule-based system which knows the intent of the failed action,[39] plan critics,[39] and bidirectional constraint propagation and comparative reasoning which operate from a detailed model of the problem solver.[44] Execution can be restarted at fixed predefined recovery points,[42] at the point where the error was detected,[40] at a point which makes sense according to the intent of the failed action,[39] at a point discovered by moving up to the next level in the plan hierarchy,[18,43] or at a point identified by using exploration to deal with unknown areas.[43] Simply restarting the plan where failure was first detected has presently been

explored only for simple linear recovery plans.[40] In addition, recovery planning must take into account the timeframe within which recovery is possible.
If humans are involved in the monitoring and recovery process, then protocols must be developed to facilitate their interactions with the automated processes. In some cases, error recovery is possible only with operator assistance, because the robot does not know the cause of the error. In this situation, the transfer from operator back to robot is handled by requiring transfer to take place at specific points in task (i.e., beginning or end of subtasks).Using this strategy, the operator must make sure that subtasks are completed if interrupted by error detection.[42]

Recovery can involve either patching an existing plan of action with no reallocation of resources or creating a new plan which requires reallocation of the subtasks to the team members. Only the latter alternative requires dynamic task allocation.

Dynamic Task Allocation

Task allocation is concerned with the distribution of tasks among the resouces of the team. Leaving the dynamic aspect aside for a moment, task distribution encompasses two issues, decomposition and assignment. Decomposition addresses the question of how a task can best be partitioned into its component parts. Quite often this is done before the task is encountered by an off-line planning process. Assignment addresses the question of how the task components can be best assigned to the team members. Assignment can be done either statically before the task starts or dynamically as work on the task is in process. Task assignment has been most directly approached by research into scheduling of automated resources. Dynamic restructuring of a team becomes important when the task involves uncertain, incomplete, or incorrect information.[4] However, the present state of planning technology does not provide much support for handling dynamic allocation in uncertain situations. Further, there is presently no unified theory on dynamic allocation even in simple cases (e.g., computing resource allocation).[4]

Design Decisions

As mentioned previously, the design of any aspect of dynamic allocation depends upon the task and, particularly, upon the extent of the designer's understanding of the task. This involves an understanding of the prevailing task phenomena and the importance of the various goals of the task. As an example of different task types, the designers of a DPS system for air traffic control include information gathering through sensing or communication input, information distribution through communication output, planning, plan evaluation, plan fixing, and plan execution.[52]

The extent of dynamic allocation which is possible and needed depends upon how much decision making is allocated to the designer, to the programmer (after design but before the task starts), to the participating humans (after task starts), and to the participating robots (after task starts). A task which is completely understood by the designer or the programmer needs no dynamic allocation. The ability to restructure the team becomes more and more

important as the task becomes less and less understood by nonparticipants. Thus, the designer must ask, "How much information about the nature and status of the task is available at design time, before the task starts and after the task starts?" If a decision must be made about allocation between humans and robots during the task then one must also ask, "How much information is available to the humans and to the robots?" If sufficient information is available after design time but before the task starts, then a number of decisions can be made and implemented through preprogramming. In cases where the programmer is omniscient of the task circumstances, then all structure and responses can be preprogrammed before the team is deployed. However, this may seldom be the case so that only team structure can be preprogrammed. This involves decisions of organization (hierarchy or committee), restructuring metaphor (commanded or negotiated), and responsibility (who is the boss and can he be trusted all the time or must that also be flexible).

For a poorly understood task the following inequality is suggested. In terms of effectiveness, reliability, and cost,

design < programming < dynamic allocation

where design determines system structure, embedded knowledge, embedded actions, embedded interactions, and execution monitoring. In uncertain situations, hard-wired design is always the least effective and reliable option but is also always the cheapest option. Similarly, dynamic allocation provides the most effective and reliable option (assuming, of course, that the dynamic allocation capabilities are well designed) but is also the most expensive in terms of design cost, implementation cost and communications and computing costs during run-time.

Another design issue related to dynamic allocation is the level of restructuring which is to be possible or the granularity of the system. The granularity should be decided by the modifiability of the system required. System granularity should always be equal or below that which is dynamically restructured. No dynamic modification should be below the level of system granularity because of undesirable consequences which arise from competing structures. In addition, the granularity of overlapping capability must match in all components which share responsibility. This has great impact upon task sharing between humans and robots and means that humans can probably always share robots' tasks, but robots cannot always share humans' tasks.

The degree of dynamic restructuring or system flexibility required is strongly dependent upon the amount of uncertainty in the task. Part of this uncertainty is manifested by component failures. In the face of failures and uncertainty, the designer must consider the consequences of only partially accomplishing the task and then prioritize the task goals. Only then can capability be allocated appropriately. This, of course, requires knowledge of what failure modes are most likely and which will have the most debilitating effects.

Other design issues abound with regards to dynamic allocation. One of the hardest relates to resolving the fuzzy area between the overlap of human and robot capabilities. The designer

must resolve the allocation decisions of possible system structures, dynamic allocation technique, allocation interaction types, and restructuring and allocation criteria. The criteria for success depend upon the uncertainty of the situation, the allocation algorithm, and the definition of optimality.

Opportunistic Scheduling

Resources should be allocated to a task at run-time to ensure best system performance, because the exact process time schedules often cannot be foreseen.[28] In addition, dynamic resource allocation improves efficiency, reliability, and flexibility[27,28] and ensures conflict-free operation of multiple robots in a dynamic environment. Opportunistic scheduling also improves the ability of robots to cooperate in a task and makes it possible for robots to share resouces.[27] Furthermore, the issue of distributed operation is emphasized because a global centrally issued allocation strategy becomes more costly as complexity increases.[24]

Most if not all existing opportunistic scheduling schemes start with a task plan which has been developed off-line.[24–31] Three levels of coordination have been suggested for flexible assembly: planning, resource allocation and coordinated motion planning.[24] In systems which benefit from opportunistic scheduling, a planner or programmer works with virtual resources and the scheduler maps these virtual resources into the real resources at run-time. The scheduler does gross motion planning, material routing, parts presentation scheduling, and resource allocation.[28]

Plan Representations

Several different plan representations have been suggested including simple trees,[5,28] action formalisms,[26] procedural networks,[50] partially ordered graphs,[25,31] AND/OR graphs,[33] and sparse hypergraphs,[33] For assembly planning, partially ordered graphs cannot retain all possible assembly sequences. AND/OR graphs do retain all possible assembly sequences, but exponentially growing complexity makes generating them all impractical. In addition, only a small number of the sequences will be encountered in any real situation. Thus, one suggestion is to generate a few preferred sequences off-line and infer the rest on-line.[33] This implies that on-line planning must supplement the opportunistic scheduling mechanism. In this technique, the assembly plan is represented as a hierarchical hypergraph, and frames represent domain objects and relations/actions.[33]It has been suggested that planning is more efficient if the representation is limited.[50]

Techniques

Several opportunistic scheduling or dynamic resource allocation techniques have been suggested,[24,25,27,28,31,33] and the specifics of only a few will be reviewed here as examples of current work. The field is still rapidly evolving, and further techniques are undoubtedly being developed even as this paper is being written.

Scheduling consists essentially of mapping resources onto tasks through various criteria. Most obviously, tasks cannot be allocated to resources which are unavailable, but beyond that

considerable scheduling latitude exists. Clearly, the scheduler must keep the schedule executionally feasible.[25] Knowledge-based techniques and heuristics have been used to deal with the complexities of execution uncertainties.[31] One common heuristic which is used is the principle of least commitment.[25,26] Scheduling can be goal or data directed and can use strategies which are intentional (i.e., action before sensing) or opportunistic (i.e., sensing before action) or a combination of both.[33] A scheduler which uses a combination of allocation strategies can change modes when a solution is not being generated within a reasonable time.[33]

The virtual resources of a plan can be mapped onto the real resources of a multiple robot system by searching an allocation tree for the best solution using various search strategies. In this search the scheduler can allocate a resource directly to the task, suspend a task until a resource is available, transfer a resource from one robot to another (e.g., end effector tool), or issue an error message if no solution is available to the task.[28] Another dynamic allocation technique assigns a resource supervisor to each class of resource which then synchronizes the access to that resource according to priority and eligibility. This mechanism also prevents collisions dynamically by using a resource supervisor to exclude mutual access to the swept volume of each concurrent robot motion.[27]

In yet another approach, a decentralized allocation strategy is used for individual robots. This technique uses dynamic local policies consisting of weighting factors (called allocation pressures) which are attached to each allocation goal (as opposed to decision rules) to maintain global coherence. Three allocation pressures are elucidated: production (assures assembly of low level subsystems first), coordination (assures proper task assignments when two or more robots are required to complete an assembly), and consumption (encourages robots to assume higher level assemblies for work-in-progress inventories and permits work flows to remain balanced). Allocation pressures are propagated through weighting functions. Robots are constantly evaluating the task mapping (including deallocation costs if already occupied) thus allowing a robot to take on a task more highly rated than the one with which it is engaged. Concurrency problems are avoided by making only one allocation decision at a time (through locking). A global but distributed blackboard is used for coordinating communication. The combinations of allocation pressures, resource constraints and opportunistic decisions are made by the individual robots.[24]

Dynamic allocation has also been approached by those researching DPS.[4,5,50,52] In many of these schemes, each agent is permitted to perform its own local planning using such mechanisms as a hierarchical planning paradigm,[50] deduction,[5] and heuristic scheduling rules.[52] In cases where the problem space is large, dynamic allocation can be guided by a focussing agent or by an agenda mechanism.[4]

Conflict Resolution/Negotiation

The distribution of tasks between independent agents implies the potential for conflicts to arise. One approach to conflict detection and resolution has been suggested although others may exist. This example will be discussed to illustrate the process of conflict detection and negotiated resolution. Plan consistency checking starts when an agent has a plan and learns

the plan of another agent or when the agent changes his own plan or when the agent believes he knows the plan of another agent and gets sensor information that indicates the other agent is not behaving consistently with that belief. If a conflict is detected, then the agent will not allocate further resources to that plan and will try to resolve the conflict until either the conflict is resolved or the agent has tried hard enough (in terms of computing resouces). If the conflict has not been resolved after the agent has tried hard enough then it will ask the other involved agent to try resolving the conflict.[52]

Allocation Decisions

Many factors affect task allocation aside from the basic issue of mapping task to resource. In the simplest of situations, these include who decides, how much time is available for the decision, how much tolerance of error is there, how much more information is available after the task starts, and if the task can be easily mapped onto one of the players with the required sensing and control capability. Once an allocation decision is made, then is it working? If not, can a better solution be found within a useful time? Since humans can be involved, then the question of "What modes of communication between humans and machines are possible?" must be answered. In addition, one must also address how subordinate to man the machine should be made and when it should be independent of human control or if it should be made independent at all (e.g., in the Iranian airliner situation, automation worked fine but humans failed). Robots can work faster and more reliably than humans, but humans can deal with task uncertainty better.

In more complex situations, even more factors affect allocation decisions. If knowledge-based elements are prevalent in the task, then how much they can decide the state of the task must be determined. Overt opposition to obtaining the task goals makes the allocation decision even more complex by introducing plan sabotage and agent attrition. Finally, safety is a very important and often neglected issue. How must humans and machines be protected from the task and from each other?

Allocation Criteria

Several questions arise related to the choice of allocation criteria. For instance, what criteria are used to decide when and where task allocation is necessary? Further, what are the criteria to decide when not enough information is available to decide? What resources should be risked when the choices are limited and the stakes are high? How much time is available to make allocation decisions? Sometimes how the environment affects the capability determines how much capability must be allocated (e.g., processing in high radiation environments). How does uncertainty in sensor data, in planning and allocation and in action affect the plan?

DPS for the air traffic control domain provides one example, among many possible examples, of dynamic allocation criteria. In this example, tasks are driven by states or changes in the knowledge base. A task may run to completion (thereby meeting completion criteria), run to the end of its assigned resources (at which time it is reevaluated and rescheduled), or be eliminated because of changes which invalidate it.[52]

Errors and Recovery

Finally, every allocation decision may not result in successful task completion because errors arise. These errors may be recognized by simulating the task plan and allocation scheme before implementation or by monitoring the execution of the plan. Whichever technique is used, the effects of errors must be carefully analyzed to determine their impact. Many errors will not have a signficant impact or may result in minor reallocation, but a few errors could have disasterous impact and ruin the day. These must be recognized and avoided or confronted directly. The degree of preparation for an error depends upon the probability that that error will occur and the impact the error will have upon the performance of the system. Once allocation is complete and an error occurs, then is there enough time to reallocate? Is a graceful abort possible, and when is the line between reallocation and abort crossed? In addition, how much time is available to determine what is wrong and whether to deal with it with the existing structure or to reallocate the resources?

These issues return the discussion to the topic of execution monitoring and this recursion emphasizes the intricate coupling between dynamic allocation and execution monitoring. In a cooperating team of humans and robots, one is not useful without the other. Execution monitoring identifies when and where dynamic task allocation is necessary and useful. Dynamic task allocation provides the action which makes monitoring task execution a useful process.

Conclusions

The advantages of cooperating teams of humans and robots abound, but to be successful these teams must be prepared to dynamically allocate their resources to the constantly changing demands of the tasks. Execution monitoring is required to identify when dynamic allocation must begin. A team of humans cooperating with robots can take many different forms. This decision depends strongly upon the amount of information which the designer has about the task. Static organizations are possible only when the task is well defined and component failures are improbable. Only dynamic organizations can confront most complex real world tasks with a high probability of success.

As the discussion of needed research below illustrates, the technology for implementing sophisticated teams of interacting humans and robots is quite underdeveloped. Little is understood about these systems in terms of theory, design, implementation, and use despite the support from such better developed technologies as distributed artificial intelligence, multiple cooperating robots, scheduling, error recovery and man–machine interaction.

Needed Research

Considerable research is needed in the areas of DPS, dynamic task allocation, execution monitoring, multiple robot coordination, and human–machine cooperation, to realize practical and effective symbiotic human–robot teams.

In the area of DPS, more work is needed to develop a formalized description of the categories

of control tasks at various levels of abstraction to improve human–machine communications and to aid in the multilevel abstraction of problem identification and solving. Description of control tasks in terms of mental functions would be extremely helpful to ensure effective task sharing between humans and machines. Analysis of human performance in real life situations to identify mental strategies and subjective performance criteria would also be beneficial. Methods are needed to evaluate specific interface design and system design concepts. Techniques are needed to verify the internal consistency of models. Consideration of integrated human performance which is normally studied as a collection of disjoint paradigms is also required.[53]

Further work on DPS is needed in the areas of theory, conflicting goals, organizational design, traditional distributed computing issues, expectation driven communication,[4] error detection and recovery, learning methods for error recovery and uncertainty handling.[8] Norms need to be developed for the exchange of tasks and other kinds of information between robots and between robots and humans. The effects of asynchrony, effective use of local control, use of local evaluation of progress, and building of abstractions of control knowledge also require extensive further study.[7] More work is needed to define the effects of communications strategies on coherence and performance (e.g., conflicting goals).[48] Finally, the problem of organizational self design of distributed problem solvers must be attacked.[7]

In the area of dynamic task allocation, models of human decision making and algorithms which effectively use these models are needed. More efficient dialog styles for explicit communications between humans and computers in multiple-task time-constrained situations must be identified.[54] More dynamic allocation implementation experience, and theoretical analysis of local allocation strategies, and the effects of feedback through the system are required.[24] A representation must be established for scheduling knowledge and for selecting the appropriate partial subtask orders to pass to the scheduler.[25] A formalism for allocation strategy selection and techniques for learning about when to switch strategies in dynamic situations must be developed.[33] Finally, performance analysis on the task allocation strategies of distributed problem solvers has not been adequately addressed.[4]

In the areas of execution monitoring and error recovery, the sensor signal to predicate transformation and modelling of uncertain and unreliable sensors must be better developed.[50] More effective techniques for recognizing the precise point in a task from which to continue, patch, or discontinue the plan needs further exploration.[39] Techniques are needed to manipulate more abstract goals and so more effective recovery plans can be constructed.[40]

For multiple coordinated robots, more work is needed in such areas as operating system kernals, message primitives, programming languages, and timing analysis of various implementations.[55] In human-machine cooperation, better training strategies are needed which limit human susceptibility to the combinatoric complexity present in complex systems.[56]

By no means is this list of research needs complete. It only serves as a starting point from which to address the effective design of teams of cooperating humans and robots. It also

emphasizes the distance which exists between the present state of the art and the day when such teams will be commonplace. However, this distance should not be discouraging because the benefits of the research in this area will be great and will tremendously influence developments in such related areas as multiple coordinated robots and human–machine cooperation.

References

[1] B. Chandrasekaran, "Natural and Social System Metaphors for Distributed Problem Solving: Introduction to the Issue," IEEE Trans. on Systems, Man, and Cybernetics, SMC-11 (1), January 1981, pp1–5.

[2] V. R. Lesser & D. D. Corkill, "Functionally-Accurate, Cooperative Distributed Systems," IEEE Trans. on Systems, Man, and Cybernetics, SMC-11 (1), January 1981, pp81–96.

[3] J-Y. D. Yang et al., "An Architecture for Control and Communications in Distributed Artificial Intelligence Systems," IEEE Trans. on Systems, Man, and Cybernetics, SMC-15 (3), May/June 1985, pp316–326.

[4] K. S. Decker, "Distributed Problem-Solving Techniques: A Survey," IEEE Trans. on Systems, Man, and Cybernetics, SMC-17 (5), September/October 1987, pp729–740.

[5] K. Konolige & N. J. Nilsson, "Multiple-Agent Planning Systems," Proc. of the National Conf. on Artificial Intelligence, Stanford, CA, 18–21 August 1980, pp138–141.

[6] W. A. Kornfield & C. Hewitt, "The Scientific Community Metaphor," IEEE Trans. on Systems, Man, and Cybernetics, SMC-11 (1), January 1981, pp24–33.

[7] N. S. Sridharan, "1986 Workshop on Distributed AI," AI Magazine, 8 (3), Fall 1987, pp75–85.

[8] A. J. Koivo & G. A. Bekey, "Report of the Workshop on Coordinated Multiple Robot Manipulators: Planning, Control, and Applications," IEEE Jour. of Robotics and Automation, RA-4 (1), February 1988, pp91–93.

[9] E. Freund, "On the Design of Multi-Robot Systems," Proc. of the IEEE Int. Conf. on Robotics, Atlanta, GA, 13–15 March 1984, pp477–490.

[10] L. Gouzenes, "Collision Avoidance for Robots in an Experimental Flexible Assembly Cell," Proc. of the IEEE Int. Conf. on Robotics, Atlanta, GA, 13–15 March 1984, pp474–476.

[11] J. Roach & M. Boaz, "Coordinating the Motions of Robot Arms in a Common Workspace," Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis, MO, 25–28 March 1985, pp494–499.

[12] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis, MO, 25–28 March 1985, pp500–505.

[13] E. Freund & H. Hoyer, "Pathfinding in Multi-Robot Systems: Solution and Applications," Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 7–10 April 1986, pp103–111.

[14] P. Tournassoud, "A Strategy for Obstacle Avoidance and Its Application to Multi-Robot Systems," Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 7–10 April 1986, pp1224–1229.

[15] Y. P. Chien et al., "On-Line Generation of Collision Free Trajectories for Multiple Robots," Proc. of the IEEE Int. Conf. on Robotics and Automation, Philadelphia, PA, 24–29 April 1988, pp209–214.

[16] R. Alami, "NNS: A LISP-Based Environment for the Integration and Operating of Complex Robotics Systems," Proc. of the IEEE Int. Conf. on Robotics, Atlanta, GA, 13–15 March 1984, pp349–353.

[17] M. Herman & J. S. Albus, "Overview of the Multiple Autonomous Underwater Vehicles (MAUV) Project," Proc. of the IEEE Int. Conf. on Robotics and Automation, Philadelphia, PA, 24–29 April 1988, pp618–620.

[18] R. J. Norcross, "A Control Structure for Multi-Tasking Workstations," Proc. of the IEEE Int. Conf. on Robotics and Automation, Philadelphia, PA, 24–29 April 1988, pp1133–1135.

[19] H. S. Baird et al., "Coordination Software for Robotic Workcells," Proc. of the IEEE Int. Conf. on Robotics, Atlanta, GA, 13–15 March 1984, pp354–360.

[20] G. Johannsen & W. B. Rouse, "Studies of Planning Behavior of Aircraft in Normal, Abnormal, and Emergency Situations," IEEE Trans. on Systems, Man, and Cybernetics, SMC-13(3), May/June 1983, pp267–278.

[21] B. G. Silverman, "Distributed Inference and Fusion Algorithms for Real-Time Supervisory Controller Positions," IEEE Trans. on Systems, Man, and Cybernetics, SMC-17 (2) March/April 1987, pp230–239.

[22] K. Niwa, "A Knowledge-Based Human-Computer Cooperative System for Ill-Structured Management Domains," IEEE Trans. on Systems, Man, and Cybernetics, SMC-16 (3), May/June 1986, pp335–342.

[23] O. Dunkler et al., "The Effectiveness of Supervisory Control Strategies in Scheduling Flexible Manufacturing Systems," IEEE Trans. on Systems, Man, and Cybernetics, SMC-18 (2), March/April 1988, pp223–237.

[24] L. Gasser & G. Bekey, "Task Allocation among Multiple Intelligent Robots," Proc. of the 1987 Workshop on Space Telerobotics, Vol. 1, G. Rodriguez, ed., Jet Propulsion Laboratory, Pasadena, CA, 1 July 1987, pp127–130.

[25] B. R. Fox & K. G. Kempf, "Opportunistic Scheduling for Robotic Assembly," Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis, MO, 25–28 March 1985, pp880–885.

[26] M. J. P. Shaw & A. B. Whinston, "Automatic Planning and Flexible Scheduling: A Knowledge-Based Approach," Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis, 25–28 March 1985, pp890–894.

[27] O. Z. Maimon, "A Multi-Robot Control Experimental System with Random Parts Arrival," Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis, MO, 25–28 March 1985, pp895–900.

[28] R. Cassinis, "Automatic Resource Allocation in Industrial Multirobot Systems," Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 7–10 April 1986, pp2012–2017.

[29] O. Z. Maimon & S. B. Gershwin, "Dynamic Scheduling and Routing for Flexible Manufacturing Systems That Have Unreliable Machines," Proc. of the IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 31 March–3 April 1987, pp281–288.

[30] R. Akella & B. Krogh, "Hierarchical Control Structures for Multi-Cell Flexible Assembly System Co-Ordination," Proc. of the IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 31 March–3 April 1987, pp295–299.

[31] B. R. Fox & K. G. Kempf, "Reasoning about Opportunistic Schedules," Proc. of the IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 31 March–3 April 1987, pp1876–1882.

[32] B. H. Krogh & R. S. Sreenivas, "Essentially Decision Free Petri Nets for Real-Time Resource Allocation," Proc. of the IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 31 March –3 April 1987, pp1005–1011.

[33] X. Xia & G. A. Bekey, "SROMA: An Adaptive Scheduler for Robotic Assembly Systems," Proc. of the IEEE Int. Conf. on Robotics and Automation, Philadelphia, PA, 24–29 April 1988, pp1282–1287.

[34] O. Berman & O. Maimon, "Cooperation among Flexible Manufacturing Systems," IEEE Jour. of Robotics and Automation, RA-2 (1), March 1986, pp24–30.

[35] C. L. Chen et al., "Task Assignment and Load Balancing of Autonomous Vehicles in a Flexible Manufacturing System," Proc. of the IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 31 March–3 April 1987, pp1033–1039.

[36] M. R. Genesereth, "Diagnosis Using Hierarchical Design Models," Proc. of the National Conf. on Artificial Intelligence, Pittsburgh, PA, 18–20 August 1982, pp278–283.

[37] R. Davis et al., "Diagnosis Based on Description of Structure and Function," Proc. of the National Conf. on Artificial Intelligence, Pittsburgh, PA, 18–20 August 1982, pp137–142.

[38] M. Gini & R. Smith, "Monitoring Robot Actions for Error Detection and Recovery," Proc. of the 1987 Workshop on Space Telerobotics, Vol. 3, G. Rodriguez, ed., Jet Propulsion Laboratory, Pasadena, CA, 1 July 1987, pp67–78.

[39] M. Gini et al., "The Role of Knowledge in the Architecture of a Robust Robot Controller," Proc. of the IEEE Int. Conf. on Robotics and Automation, St. Louis, MO, 25–28 March 1985, pp561–567.

[40] R. E. Smith & M. Gini, "Robot Tracking and Control Issues in an Intelligent Error Recovery System," Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 7–10 April 1986, pp1070–1075.

[41] M-Y. Chern, "An Efficient Scheme for Monitoring Sensory Conditions in Robot Systems," Proc. of the IEEE Int. Conf. on Robotics, Atlanta, GA, 13–15 March 1984, pp298–303.

[42] M. M. Moya & W. M. Davidson, "Sensor-Driven Fault-Tolerant Control of a Maintenance Robot," Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 7–10 April 1986, pp428–434.

[43] B. Ward & G. McCalla, "Error Detection and Recovery in a Dynamic Planning Environment," Proc. of the National Conf. on Artificial Intelligence, Pittsburgh, PA, 18–20 August 1982, pp172–175.

[44] E. Hudlicka & V. R. Lesser, "Meta-Level Control through Fault Detection and Diagnosis," Proc. of the National Conf. on Artificial Intelligence, Austin, TX, 6–10 August 1984, pp153–161.

[45] B. R. Donald, "Planning Multi-Step Error Detection and Recovery Strategies," Proc. of the IEEE Int. Conf. on Robotics and Automation, Philadelphia, PA, 24–29 April 1988, pp892–897.

[46] B. T. Mitchell & D. I. Kountanis, "A Reorganization Scheme for a Hierarchical System of Learning Automata," IEEE Trans. on Systems, Man, and Cybernetics, SMC-14 (2), March/April 1984, pp328–334.

[47] R. G. Smith & R. Davis, "Frameworks for Cooperation in Distributed Problem Solving," IEEE Trans. on Systems, Man, and Cybernetics, SMC-11 (1), January 1981, pp61–70.

[48] E. H. Durfee et al., "Increasing Coherence in a Distributed Problem Solving Network," Proc. of the 9th Int. Joint Conf. on Artificial Intelligence, Los Angeles, CA, 18–23 August 1985, pp1025–1030.

[49] R. P. Sobek, "A Robot Planning Structure Using Production Rules," Proc. of the 9th Int. Joint Conf. on Artificial Intelligence, Los Angeles, CA, 18–23 August 1985, pp1103–1105.

[50] D. E. Wilkins, "Recovering from Execution Errors in SIPE," Computational Intelligence, 1 (1), February 1985, pp33–45.

[51] M. Georgeff, "A Theory of Action for MultiAgent Planning," Proc. of the National Conf. on Artificial Intelligence, Austin, TX, 6–10 August 1984, pp121–125.

[52] D. McArthur et al., "A Framework for Distributed Problem Solving," Proc. of the National Conf. on Artificial Intelligence, Pittsburgh, PA, 18–20 August 1982, pp181–184.

[53] J. Rasmussen, "Skills, Rules and Knowledge; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models," IEEE Trans. on Systems, Man, and Cybernetics, SMC-13 (3), May/June 1983, pp257–266.

[54] J. S. Greenstein & M. E. Revesman, "Two Simulation Studies Investigating Means of Human-Computer Communication for Dynamic Task Allocation," IEEE Trans. on Systems, Man, and Cybernetics, SMC-16 (5), September/October 1986, pp726–730.

[55] K. G. Shin & M. E. Epstein, "Intertask Communications in an Integrated Multirobot System," RA-3 (2), April 1987, pp90–100.

[56] J. Sharit, "The Use of Measures of Entropy in Evaluating Human Supervisory Control of a Manufacturing System," IEEE Trans. on Systems, Man, and Cybernetics, SMC-17 (5), September/October 1987, pp815–820.

Human–Machine System Architecture: The Design of Cooperative Teams of Human and Computer Decision Makers

Christine M. Mitchell
Center for Human–Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332
(404) 894-4321
mitchell@chmsr.gatech.edu

December 1988

Abstract

As computer technology becomes both more sophisticated and more affordable, design issues become more challenging. In complex, dynamic systems, computers can be used to replace human decision makers, to enhance decision making via decision aiding, or to amplify human potential by providing the human operator an associate or assistant in the decision making process. In complex systems, uncertainty associated with system dynamics and the costs and risks associated with system malfunction make the first option, replacement of the human decision maker with a computer decision maker, an unrealistic alternative. Recent research has shown that display enhancement is a design alternative that has proved successful in reducing operator workload and increasing operator effectiveness. This paper describes research that examines the feasibility and effectiveness of the third alternative—design teams of decision makers that include both humans and computers. This design strategy is sometimes called an operator's associate. The function of an associate is to provide intelligent assistance for the human operator of a complex dynamic system. The basis for intelligent, context-sensitive advice and reminders is the ability of the associate to infer likely operator intentions in real time. Such human-computer architectures draw heavily on artificial intelligence models. This paper discusses the general requirements for an operator's associate and presents an overview of three components of an on-going research project: OFMspert (Operator Function Model expert system), a particular architecture of an intelligent operator's associate; ACTIN, OFMspert's intent-inferencing mechanism; and Ally, an implementation of OFMspert with system control functions.

Key Words: Human–machine interaction, operator's associate, intelligent decision aiding, supervisory control systems.

Introduction

The increasing availability and sophistication of computer technology has contributed to its proliferation in work environments. Increased automation has brought about major changes in the role of the human operator in the control of a complex dynamic system. In particular, the human's role has shifted from that of a manual controller, where perceptual-motor skills are emphasized, to a supervisory controller, where cognitive skills such as planning, monitoring, and decision making are emphasized (Baron, 1984; Rasmussen, 1986; Sheridan et al., 1988; Wickens, 1984; Woods, 1986).

The consequences of automating a complex dynamic system, and thus changing the role of the human to that of a supervisory controller, are often problematic. In particular, Wickens (1984) cites several problems: an increased monitoring load; a "false sense of security" whereby the operator trusts the automation to such an extent that any human intervention or checking seems unnecessary; and "out-of-the-loop familiarity" (i.e., as a supervisory controller who no longer actively controls the system, failure detection, diagnosis, and compensation can take a very long time). These and other difficulties have serious implications for the ability of operators to cope with emergency situations. However, as Chambers and Nagel (1985) point out, at present, humans are an irreplaceable part of complex systems. They note that humans are capable of coping with ambiguity and uncertainty and can make inductive decisions in novel situations.

Given that the human will remain an integral part of a complex system, a potential approach to advanced automation is that of "amplifying" rather than automating human skills (Woods, 1986). Thus, the computer acts as an assistant to the operator, capable of performing routine control tasks, monitoring human performance, and offering advice. In this way the operator is "still in the loop." Chambers and Nagel suggest that this approach may lead to better human-system performance and enhanced user acceptance.

Computerized advice giving is a particularly important consideration for intelligent aiding and decision support. In Vicente and Rasmussen's (1987) taxonomy of decision support techniques, the authors advocate a blend of approaches that enables the operator to understand the functions of the automated aid. The human operator remains in control of the system, with an automated assistant that integrates data, matches the problem representation employed by the user, and gives advice and supporting explanations. Similarly, Wickens (1984) argues that to promote user acceptance, a cooperative computer system is necessary. Such a system should perform tasks consistent with the human operator, be "flexibly available," and should provide recommendations rather than commands. Woods (1986) supports the idea of a "joint cognitive system" where the computer aids the user in the process of decision making rather than merely presenting a solution or recommendation. Thus, the computer consultant is a resource for the human operator; the human, however, is still in charge. These ideas form the design philosophy of the operator's associate.

An Operator's Associate

The concept of a computer-based operator's assistant or associate has been proposed as one method to provide an intelligent decision aid for operators responsible for complex dynamic systems (Chambers and Nagel, 1985; Rouse et al., 1987; Rubin et al., 1988a). An operator's associate is a computer-based system that acts as an assistant to the human operator. Functionally, an operator's assistant can offer the operator timely advice and reminders, and, at the operator's request, assume responsibility for portions of the supervisory control task. The principle is to augment a team of human operators with one or more computer-based assistants. A human-computer team achieves the goals of improved system efficiency and safety. System efficiency is achieved by simultaneously reducing the number of required human operators and incorporating state-of-the-art knowledge-based computational techniques. System safety and emergency backup are achieved by retaining the human in the control system to compensate for unexpected events and computer system limitations (Wickens, 1984).

A human-computer team may address some of the current problems with supervisory control systems. As systems become more automated, the human operator performs fewer tasks on a routine basis. In complex dynamic systems, however, safety requires staffing at a level that can meet the most challenging or threatening abnormal conditions. The result is often a team of human operators who are rarely challenged and often underutilized. Human factors research suggests that boredom and underutilization may actually impair operator effectiveness in abnormal or emergency conditions. A human-computer cooperative team may reduce some of the tedium. A cooperative system entails addition of a computer-based decision maker and a concurrent reduction of human personnel to the number of operators that, in cooperation with the computer-based system, can adequately monitor and control the system under a wide range of system conditions. Consolidation of decision making into a team comprised of human operators augmented by a computer-based assistant retains the human operator as an essential part of the decision making team, yet gives the operator an assistant to whom tedious tasks can be delegated under normal conditions or who can assume responsibility for lower priority tasks under abnormal, i.e., high workload, conditions.

An operator's associate (or, in the case of aviation, a pilot's associate) (e.g., Chambers and Nagel, 1985; Rouse et al., 1987) is a fascinating concept. Before implementation, however, a range of issues must be addressed. These include the types of functions that the associate can perform, the relationship between the human and computer decision makers (e.g., who has control or initiative during normal or emergency events), and the interaction between the human and computer components (e.g., how or when they communicate intentions and assumptions to each other). There are many ways that these issues can be addressed and implemented. At this time there is not a correct or best way. Precise specification of the characteristics of an operator's associate are open research questions that require principled implementation and empirical evaluation. This paper describes a particular conceptual design for an operator's associate. This design is part of a larger research project that attempts to build a theory of human-computer interaction in the control of complex dynamic systems. To evaluate the theory, individual portions are implemented and empirically

evaluated. This paper summarizes some basic characteristics that underlie our operator's associate design. Then, we summarize the structures that provide the computer with knowledge about the system and operator functions and with the ability to understand the operator's current intentions and goals. Finally, we describe the procedures and results of two empirical studies: one that evaluated the associate's understanding capabilities and the second that evaluated the effectiveness of a human-computer team using the associate.

Principles of an Operator's Associate

Conceptually, an operator's associate is a dynamic entity with which the operator can interact with the ease and familiarity of human-to-human interaction. The intent is to design the computer component of the supervisory control system so that it mimics the functions that a human assistant performs. The computer component's knowledge base and user interaction should permit the system to swiftly assume responsibility for control tasks that the human operator may delegate and to offer the human operator context-sensitive suggestions, advice, and reminders.

As a software system, the operator's associate is a stand-alone "expert system" that can carry out system control functions under the anticipated or planned set of system conditions and events. As part of a decision-making team, the operator's associate is designed to function as a subordinate in the decision-making process. The associate monitors the system and may offer advice or reminders, but it only assumes control of portions of the system when the operator explicitly delegates responsibility for specific activities.

The subordinate role of the operator's associate is a fundamental assumption that characterizes our research project. The rationale for this is that in complex dynamic systems it is impossible to anticipate and plan for all contingencies (Rasmussen and Goodstein, 1987). Thus, a computer system cannot act as the principal or sole "expert" in system control. A human decision maker will always be present and ultimately responsible for effective and safe system operation. In order to provide the operator with the knowledge and authority necessary to intervene and control the system during unexpected events, the control system must be a cooperative system in which computer-based tools and systems are used to enhance rather than replace human decision-making functions (Woods, 1986; Rasmussen and Goodstein, 1987). By defining the associate as a stand-alone system, the capabilities of knowledge-rich computational models such as expert systems can be used to enhance the effectiveness of system control. By building a cooperative system in which the human is the primary decision maker, system design ensures that the human responsible for system efficiency and safety has the necessary knowledge and authority.

The interactive nature of the cooperative control team implies a dynamic allocation of control functions between the human and computer components. Both the human and computer are able to perform any control activity. Cooperative design, as proposed in this design for an operator's associate, has advantages and costs. The design forecloses the

possibility of optimizing control system function by allocating individual tasks and functions to the entity "most" capable of carrying them out. It does so, however, to insure that the human operator is integrated into every level of the supervisory control system and is able to carry out any of the control activities if necessary.

Dynamic task allocation allows the human operator to prioritize system control activities and take advantage of the computer's strengths and compensate for the computer's weaknesses in the context of current system state. The human operator, in working with the computer-based controller, can build an understanding and trust for the range of activities that the computer handles well, and likewise build an understanding of situations when the computer fails to understand the nuances of an event or fails to recognize the occurrence of a novel event. Allowing the operator to build up a thorough understanding of the computer system's strengths and weaknesses is an essential part of the cooperative supervisory control system design.

The intelligence and utility of the operator's associate rest on its abilities to understand the operator's current intentions and to provide context-sensitive assistance in the form of operator aids (e.g., suggestions, advice, or reminders) or by assuming responsibility given for portions of the control task. To ensure generalizability, the operator's associate requires a well-defined knowledge structure that represents information about the controlled system and the operator functions, as well as a problem-solving structure to build a dynamic representation of operator intentions which reflects current system state and recent operator actions. There are many candidate models that might be used for both of these requirements (Geddes, 1985; Jones, 1988). This project uses the operator function model (OFM) (Mitchell, 1987) to organize knowledge about the controlled system and related operator functions, and the blackboard model of problem solving (Nii, 1986) to build a current hypothesis of operator intentions. The next section summarizes how these models are used in ACTIN, the understanding component of OFMspert (Operator Function Model expert system).

ACTIN (Actions Interpreter): OFMspert's Understanding Component

OFMspert is a research project that explores the design of an operator's associate for complex dynamic systems (Rubin et al., 1988a). It uses the operator function model (OFM) as the knowledge structure to represent operator intentions. OFMspert is a generic architecture for an operator's associate. It specifies the individual components necessary to communicate with the controlled system, to construct representations of controlled system state and current operator intentions, and to coordinate the timing and communication between these modules. The details of the architecture are given in Rubin et al. (1988a, 1988b). This paper focuses on ACTIN, the OFMspert component that performs operator-intent inferencing. While Rubin et al. (1988a) show that the architecture as a whole is feasible, the next step is to validate that the intent inferencing architecture is also "correct"; i.e., ACTIN can build a representation of current operator intentions that is consistent with what current operator intentions actually are. ACTIN uses the problem solving capabilities of a blackboard and the

system and operator knowledge derived from an OFM to build and maintain a current hypothesis about operator intentions given current system state. First, we summarize the main features of the OFM and the blackboard model of problem solving; then we show their application in ACTIN.

Operator Function Model

In previous research, the OFM provided a flexible framework for representing operator functions in the context of a dynamic system (Mitchell and Saisi, 1987). The OFM is a representation of how an operator might decompose and coordinate system control functions. Mathematically, the OFM is a hierarchic-heterarchic network of finite-state automata. Network nodes represent operator activities defined as operator functions, subfunctions, tasks, and actions. The network is a hierarchy that represents operator functions as the highest level nodes and decomposes individual operator functions into subfunctions, tasks, and actions. Actions may be manual or cognitive. Manual actions consist of one or more system reconfiguration commands. Cognitive actions include information gathering, information processing, and decision making; typically, cognitive actions are supported by information retrieval requests. Network arcs represent system events or the results of operator actions that initiate or terminate operator activities. At each level of the hierarchy, there may be a heterarchy, i.e., a collection of activities that, given the corresponding system events, are undertaken concurrently. The heterarchy account for the coordination of operator activities and the operator's dynamic focus of attention.

Figure 1 depicts an abstract OFM. The highest level consists of five operator functions. Each function decomposes into multiple subfunctions; likewise, subfunctions decompose into tasks and tasks into actions. One feature of the OFM is that subfunctions, tasks, or actions may be performed in support of more than one higher-level activity; which one(s) depends on current system state. For example, in Figure 1, action 2 may be undertaken (or performed) to support either task 2 or 3, or possibly both. This context-sensitivity is an important attribute in applying the OFM to intent inferencing.

The OFM is a prescriptive model that specifies nondeterministically a set of plausible operator functions, subfunctions, tasks, and actions given current system state and recent operator actions. As such, it provides a structure to represent knowledge about the system and operator function and a mechanism to define expectations of operator activities given current state. For example, in considering the OFM depicted in Figure 1, if system event a occurred, it is reasonable to expect operator function 2 to become active.

Blackboard Model of Problem Solving

In general, a model for problem solving "provides a conceptual framework for organizing knowledge and a strategy for applying that knowledge" (Nii, 1986). A problem-solving model specifies a scheme for knowledge representation and a control strategy in order to construct a solution to the problem. ACTIN uses the HASP blackboard model of problem solving (Nii et al., 1982). The HASP blackboard is one of the few artificial intelligence systems that explicitly addresses real-time hypothesis formation in dynamic environments.

The blackboard model of problem solving consists of three components: the blackboard, the knowledge sources, and the blackboard control (Nii, 1986). The blackboard is a data structure on which the current best hypothesis of the solution is maintained and modified. The hypothesis is represented hierarchically, at various levels of abstraction, and evolves incrementally over time as new data become available or as old data become obsolete. Domain-specific knowledge to construct and update the blackboard is contained in knowledge sources. Knowledge sources are logically independent pieces of information that perform several diverse functions, e.g., post information at appropriate levels of the blackboard, connect information between levels, and assess the state of the current hypothesis. Blackboard control applies knowledge sources opportunistically with both top-down and bottom-up reasoning, depending on what is more appropriate in the current context.

As a problem-solving model, the HASP blackboard is very compatible with both the system and operator function knowledge represented in the OFM. Both models represent hypotheses hierarchically. The blackboard hierarchy offers a mechanism to maintain both heterarchic and hierarchic information about operator activities contained in the OFM nodes. The blackboard knowledge sources provide a conceptually efficient structure to maintain much of the domain-specific knowledge contained in the OFM arcs. The opportunistic control strategy offers the flexibility of both event-driven and goal-driven reasoning. ACTIN, OFMspert's intent inferencer, combines domain knowledge from the OFM and the problem-solving framework of the blackboard model to dynamically build and maintain a hypothesis about current operator intentions.

ACTIN's Structure

Like all blackboard problem-solving systems, ACTIN consists of a blackboard data structure, knowledge sources, and a blackboard control mechanism. Figure 2 illustrates ACTIN's architecture. The blackboard data structure hierarchically represents hypothesized current operator intentions as a collection of operator goals decomposed into related plans, tasks, and actions. The goal-plan-task-action representation corresponds to the function-subfunction-task-action OFM hierarchy. Goals are currently instantiated functions, plans are currently instantiated subfunctions, and so forth. In some respects, ACTIN is a process model that uses the blackboard problem-solving method to build a dynamic representation of current operator intentions based on the static information about operator activities contained in the OFM (Wenger, 1987). Figure 3 depicts the four-level hierarchy of ACTIN's blackboard.

Knowledge sources contain domain-specific knowledge to construct, update, and assess the blackboard data structure. Knowledge sources that construct and update the blackboard are of two types: model-driven and data-driven. Model-driven knowledge sources use the OFM knowledge to instantiate hypothesized goal-plan-task (GPT) structures based on system events and recent operator actions. For a system-triggering event, defined by OFM arcs, a knowledge source may post a goal, one or more plans, and associated tasks on the respective blackboard levels. The sets of goals, plans, and tasks are defined by network nodes in the OFM. For example, a system failure may cause a knowledge source to post a GPT structure that hypothesizes a new operator goal (together with its associated plans and tasks) to compensate for the observed failure. Referring to Figure 1, system event a triggers instantiation of function 2 as a goal with its related plans and tasks.

Data-driven knowledge sources post observed operator actions on the ACTIN blackboard at the lowest level of abstraction and attempt to link these actions to currently hypothesized GPT structures. By linking an observed action to one or more tasks, ACTIN infers the intent of (i.e., understands) the action. If an action is not linked, it is not currently considered to be understood. The knowledge of what actions support which tasks is derived from the OFM.

Data-driven knowledges sources have the property of maximal connectivity; they connect an action to all possible tasks that the action can support. Maximal connectivity gives ACTIN flexibility in subsequent interpretation of operator actions and the ability to maintain multiple, competing hypotheses - much like a human operator might have.

Although data-driven knowledge sources typically connect observed actions to previously hypothesized GPTs, some data-driven knowledge sources can themselves hypothesize and construct a GPT. That is, using a collection of operator actions, some knowledge source can consider the actions as a set to infer a new GPT. This capability allows ACTIN to infer operator intentions with both top-down and bottom-up strategies.

Assessment knowledge sources determine the extent to which operator actions support ACTIN's currently hypothesized operator goals, plans, and tasks. Assessments are always made in the context of a particular GPT and are the mechanism that ACTIN uses to create a context for possible advice or reminders.

Like HASP, ACTIN's blackboard control consists of a hierarchy of knowledge sources: the strategy, activators, and specialists. The strategy knowledge source examines three lists of events maintained by the blackboard in order to guide its focus of attention. After selecting a type of event on which to focus, the strategy knowledge source invokes an appropriate activator knowledge source, which in turn chooses specialist knowledge sources to process the event. Specialist knowledge sources are the only knowledge sources that can directly modify the blackboard data structure. The sets of assessment, model-driven, and data-driven knowledge sources described above comprise ACTIN's specialist knowledge sources.

The strategy knowledge source uses three lists to coordinate blackboard activity: the clock-events list, the problems list, and the events list. The clock-events list contains events scheduled for future execution. The problems list saves anomalous information (e.g., actions that were uninterpretable when they occurred). The events list contains all new events that must be processed immediately by the blackboard.

GT-MSOCC: An Application Domain

In order to test the modeling techniques and structural components of OFMspert, an OFMspert of the Georgia Tech Multisatellite Operations Control Center (GT-MSOCC) was implemented. GT-MSOCC is an interactive, real time simulation of the NASA Multisatellite

Operations Control Center (MSOCC), a NASA ground control station for near-earth satellites located at Goddard Space Flight Center in Greenbelt, MD (Mitchell, 1987). In GT-MSOCC, an operator monitors the data transmitted by satellites to insure data integrity, compensates for equipment and schedule failures, and handles system requests.

The GT-MSOCC system and the duties of the GT-MSOCC operator are described briefly below. More detail is available in Mitchell, 1987; Mitchell and Saisi, 1987; and Mitchell and Forren, 1987.

The GT-MSOCC System

GT-MSOCC supports 16 near-earth satellites and the Space Shuttle. Different satellites have different requirements for the number and types of equipment necessary to capture and process their telemetry data. In general, all satellites use several NASA communication lines (NASs) to transmit their data through a variety of computer and communication networks for data processing and recording. These configurations may include a RUP (Recorder Utility Processor), a TAC (Telemetry and Command Computer), one or more APs (Application Processors), a GW (GateWay network), a CMS (Command Management System), and a VIP (Virtual Interface Processor). Finally, data are sent to an MOR (Mission Operations Room) for satellites or an SPF (Shuttle Payload Facility) for the Space Shuttle. These are spacecraft-specific control rooms where operators monitor and control the spacecraft. Figure 4 illustrates this system. Note that RUPs, CMSs, GWs, and VIPs do not transmit data to subsequent components; rather, they are "endpoints" in the equipment configurations.

The data transmitted by GT-MSOCC satellites consist of telemetry (science) blocks and two types of error blocks (polynomial errors and sequence errors). The GT-MSOCC operator is responsible for monitoring the rates of transmission and knows about the different expected block and error rates at each point in the equipment string. The expected rate depends on the type of equipment and the particular arrangement of the entire equipment configuration for a specific satellite.

The GT-MSOCC operator workstation consists of three display monitors and one keyboard (see Figure 5). The center screen shows the hardware status and mission configuration display, as illustrated in Figure 6. This display shows the hardware status of all GT-MSOCC equipment together with currently configured missions and their supporting communication and computer equipment. For example, in Figure 6 the ERBE spacecraft is currently supported by NAS 18, NAS 21, NAS 29, RUP 2, TAC 4, AP 6, GW 1, CMS 2, VIP 1, and MOR 5.

The right display gives data transmission information. The operator can request various telemetry pages which give the data and error counts for individual pieces of equipment. The left display is used for schedule information. In addition to the MSOCC schedule page, which shows the imminent missions and their associated equipment, the operator can request schedules for entire classes of equipment (e.g., the request for the "TAC avail" page shows a

graphical representation of schedules for all TACs and APs) or individual mission or equipment schedules.

GT-MSOCC Operator Functions

At the highest level of the GT-MSOCC operator function model are major operator functions and the system events that cause the operator to transition among functions (see Figure 7). This level of description represents operator goals in the context of current system state. The arcs define system events that trigger a refocus of attention or the addition of a function to the current set of operator duties.

The default high-level function is to control current missions. This involves the subfunctions of monitoring data transmission and hardware status, detection of data transmission problems, and compensation for failed or degraded equipment. Each subfunction is further defined by a collection of tasks, which in turn are supported by operator actions (system reconfiguration commands or display requests).

System-triggering events cause the operator to focus attention on other high-level functions. An unscheduled support request causes the operator to shift to the "configure to meet support requests" function. An error message from the automatic scheduler causes the operator to transition to the function to compensate for the automatic schedule failure. A request to deconfigure a mission causes the operator to shift to the function of deconfiguring a manual mission configuration. Finally, the operator may engage in long-term planning in the absence of other system-triggering events. Upon the termination of these other functions, the operator resumes the default control of current missions function. Functions may be terminated by their successful completion or the determination that they cannot be completed.

GT-MSOCC Implementation

GT-MSOCC runs on a VAX 11/780 under the Berkeley UNIX operating system (BSD 4.3). Three InteColor 2400 display terminals and one keyboard are used for the operator interface. OFMspert was originally implemented in Smalltalk/V on a PC AT, 12 mHz machine. Currently, OFMspert runs in Smalltalk 80 on a Macintosh II. OFMspert communicates with the VAX via an RS232 communications port. This configuration is shown in Figure 8. The details of OFMspert implementation are discussed in Rubin et al., 1988a and 1988b.

ACTIN for GT-MSOCC

The organization of GT-MSOCC goals, plans, tasks, and actions are defined by the GT-MSOCC OFM (Figure 7). Given system-triggering events, ACTIN's model-driven knowledge sources post the appropriate goal, plan, and task (GPT) structures on the blackboard. When operator actions occur, ACTIN's data-driven knowledge sources post actions on the blackboard and attempt to "connect" the actions to tasks which they support. This "connection"

between actions and tasks defines ACTIN's intent inferencing capability. The knowledge of appropriate inferences of intent is contained in a data structure that matches actions to task types. Data-driven knowledge sources consult this structure to determine task type(s) that a current operator action can support. They then search the blackboard's task level of abstraction for those types and connect the action to all appropriate tasks.

A concrete example from GT-MSOCC illustrates ACTIN's dynamic construction of operator intentions. The resulting blackboard from the following scenario is shown in Figure 9. When a particular satellite (e.g., Planetary Mission [PM]) is configured automatically, ACTIN's model-driven knowledge sources post the goal to control the current mission (CCM) for PM. This goal is comprised of two plans: to monitor software (i.e., data transmission) (MSW) and to monitor hardware status (MHW). Each plan is composed of one or more tasks. The monitor software plan consists of two tasks: to check data flow at the MOR (CMOR) and to check data flow at endpoint equipment (CEND). The monitor hardware plan consists of the single task to check hardware (CHW). This entire GPT structure defines the default control of current mission function prescribed by the OFM. When PM is configured, ACTIN's knowledge sources retrieve the control of current mission GPT structure, fill in mission-specific information (e.g., the name of this particular mission is PM), and post the structure on the blackboard. Now ACTIN's blackboard contains model-derived expectations of operator actions. After some time, one of the components used by PM experiences a hardware failure. The component in this example is RUP2. Upon the occurrence of this triggering event, ACTIN's model-driven knowledge sources post a plan to replace the failed component, along with the four associated tasks of finding a currently available replacement (FCUR), finding the duration of the mission (FDUR), finding an unscheduled replacement (FUSC), and executing the replace command (XRPL). The operator then requests the schedule for RUP1. ACTIN's knowledge sources determine that a request of this type supports a "Find Unscheduled" (FUSC) type of task for RUPs, so this action is posted and connected to the FUSC task associated with the RUP2 replace plan.

The evaluation of operation performance is performed by knowledge sources that assess the degree to which operator actions support current tasks (and by extension, plans, and goals). ACTIN schedules assessments periodically in the context of particular goals or plans. In the example shown in Figure 9, ACTIN schedules separate assessments for the control of current mission goals for PM, GEO, and LNSAT, and the replace plan for RUP2. Assessments note the number of supporting actions and the time at which those actions occurred. From Figure 9, the assessment for PM would note that the CMOR task is supported by two actions and the CEND task is supported by one action. Similarly, GEO's assessment would state that two actions support the CMOR task and one action supports the CEND task. LNSAT's assessment would state that one action supports the CMOR task. RUP2's replace plan assessment would state that one action supports the FDUR task and one action supports the FUSC task.

To summarize, the proposed model for intent inferencing uses the OFM methodology to postulate operator function, subfunctions, and tasks on the basis of current system state and

observed operator actions. This model has been implemented using a blackboard architecture. This structure, of which the scenario described in this section is an example, defines the context for intent inferencing. The OFM and its implementation in ACTIN is an example of "the middle ground" in theory construction in cognitive science (Miller, Polson, and Kintsch, 1984). The theory has well-defined structures and processes to "support both the instantiation of the theory as an executable computer program and qualitative experimental studies of the theory" (Miller, Polson, and Kintsch, 1984, p. 13). The next section summarizes the procedure and experimental results for the validation of ACTIN.

Validation of Intent Inferencing

Procedure

Validation of intent inferencing assures that the system is correctly inferring the intentions of the human operator. Within the context of the OFM structure of intentions, this means that the system infers support for the same tasks (and by extension, plans, and goals) as the human, given the same set of operator actions. The "human" in this case can be a human-domain expert performing a post-hoc analysis, or the human operator giving an (on-line) account of intentions. The experimental validation of ACTIN's intent inferencing was conducted in two studies. In Experiment 1, a domain expert's interpretations of operator data were compared to ACTIN's interpretations of those same actions on an action-by-action basis. In Experiment 2, concurrent verbal protocols were collected from GT-MSOCC operators while they were controlling GT-MSOCC. Statements of intentions for each action were compared to ACTIN's interpretations. Thus, the proposed two-part framework for the validation of intent inferencing is 1) comparison of expert and ACTIN analyses; and 2) comparison of concurrent verbal protocols and ACTIN analyses (see Jones, 1988, for more detail).

For the first experiment, a domain expert hypothesized intentions from 30 hours of data from 10 GT-MSOCC operators. The data from these subjects consisted of various log files that detailed the events that occurred during the experimental sessions. Perfect state information (i.e., what missions were currently configured, what equipment failures existed) was available, as well as every action by the operator. The domain expert used these log files as the basis for interpretations.

The second approach to the validation of intent inferencing made use of verbal data as a measure of subjects' intentions in controlling GT-MSOCC. The data in Experiment 2 consist of concurrent verbal protocols from two subjects for seven GT-MSOCC sessions.

Summary of Results

Tables 1 and 2 contain a summary of experimental results. For each experiment, these tables show the percent of actions that were interpreted by ACTIN in the same way as the domain expert (Experiment 1) or the subjects in their verbal protocols (Experiment 2). Overall, ACTIN did a good job of inferring intentions for GT-MSOCC operators. Certain classes of actions were relatively straightforward to interpret and, thus, showed strong

agreement between ACTIN and human interpretations. Other classes of actions showed marked differences between ACTIN and human interpretations. In particular, some actions were interpreted as supporting tasks that were not modeled in the OFM used by ACTIN. Some mismatches can be remediated by relatively simple extensions to the GT-MSOCC OFM. Some mismatches are due to the difficulty that the OFM model has in accounting for less structured or reactive functions like browsing or long-term planning. The latter are not easily modeled, as the structure of such a function within the OFM framework is not clear. A detailed discussion of the experiments and interpretations can be found in Jones, (1988), and Jones et al., (1988).

ALLY: OFMspert With Control Properties

The most recent OFMspert research examined the use of OFMspert as the computer component of cooperative team of supervisory controllers (Bushman, 1988). OFMspert was modified to provide a user interface that allowed a human operator to request OFMspert's assistance and allowed OFMspert to communicate any advice or suggestions to the operator. In addition, OFMspert was modified to allow it to control the system at the human operator's request. Ally is the software system that was created by modifying OFMspert with a user interface and system control capabilities.

Ally Structure

Figure 10 depicts Ally's user interface. The interface was designed to support multiple operator functions at various levels of aggregation. The operator used the rectangular buttons to request assistance from Ally. Each button, when selected by the operator, displayed one or more context-sensitive submenus that allowed the operator to further specify the amount of help desired. The definitions of the buttons and the associated submenus were obtained from the GT-MSOCC OFM. The buttons themselves corresponded to the high-level operator functions. The submenus reflected associated subfunctions and tasks.

To test the effectiveness of Ally with a specific interface, an experiment was conducted in which 10 subjects controlled GT-MSOCC with both a human assistant and with Ally. The teams controlled the system for four sessions with each associate. A range of performance measures was used. The results, consistent over all measures, showed no performance differences between the teams composed of two human operators and the teams composed of a human operator and Ally. Figure 11 provides an example of the data gathered in this experiment. As the figure shows, there was a trend that indicated that performance with Ally was better than that with the two-person team; however, this trend was not statistically significant. Interestingly, although subjects liked Ally, they often said they preferred a human assistant. Also of interest are the differences in the style of interaction between the two experimental conditions. Preferences for a human or computer associate may be linked to task-oriented versus social-oriented styles of interaction (Bales, 1958).

Conclusions

The notion of a computer-based operator's associate is a conceptually appealing philosophy that has the potential to structure the design of human-computer architectures in the control of complex dynamic systems. The OFMspert research provides some initial data on design, implementation, and evaluation of this concept. Thus far, the results are encouraging. Future results will further explore the issues of human-computer communication in cooperative teams and the potential to use an operator's associate as a tutor for novice operators whose role can change into an associate or an assistant as the operator's skills evolve.

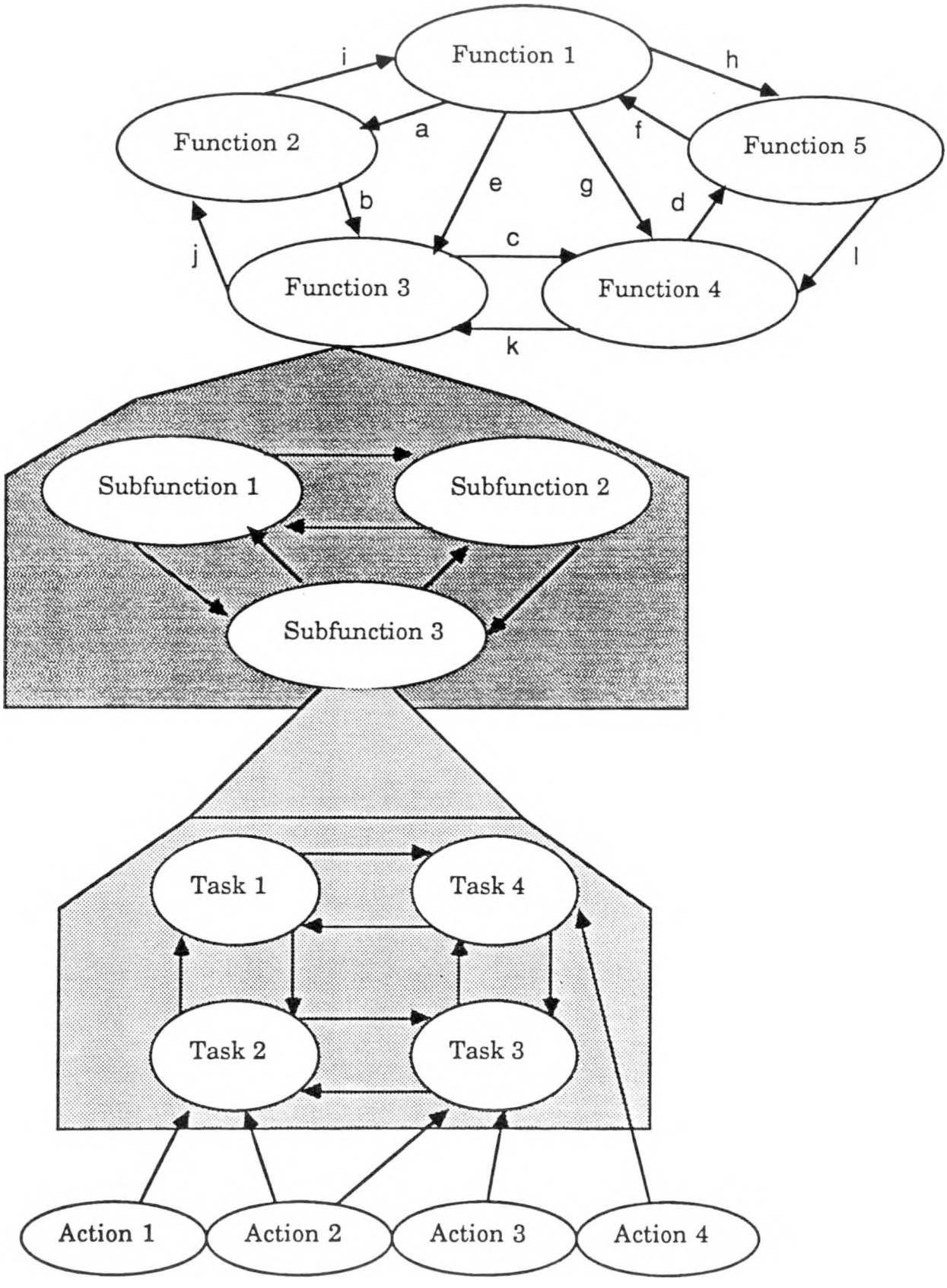


Figure 1. A Generic Operator Function Model

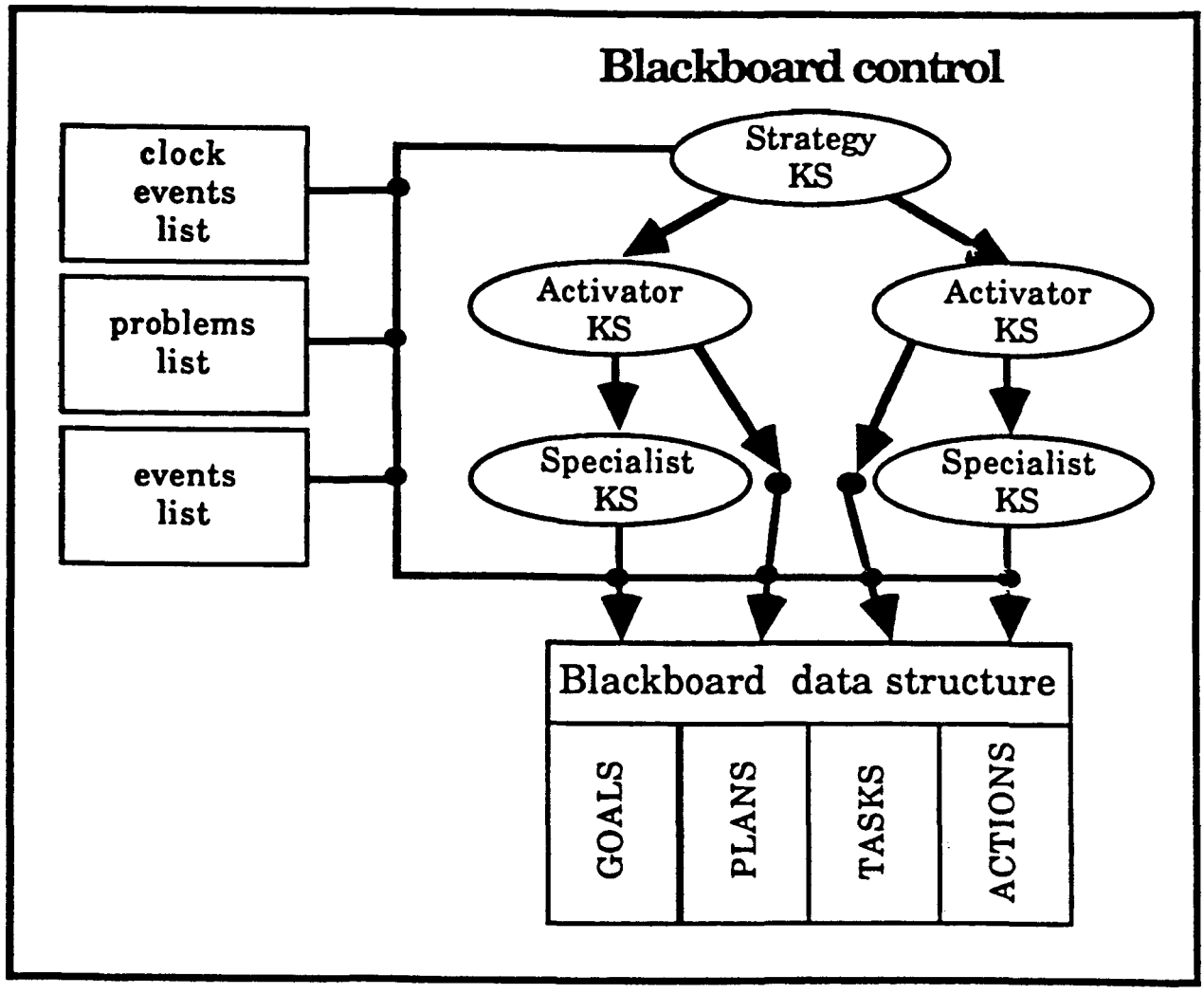


Figure 2. ACTIN's Architecture

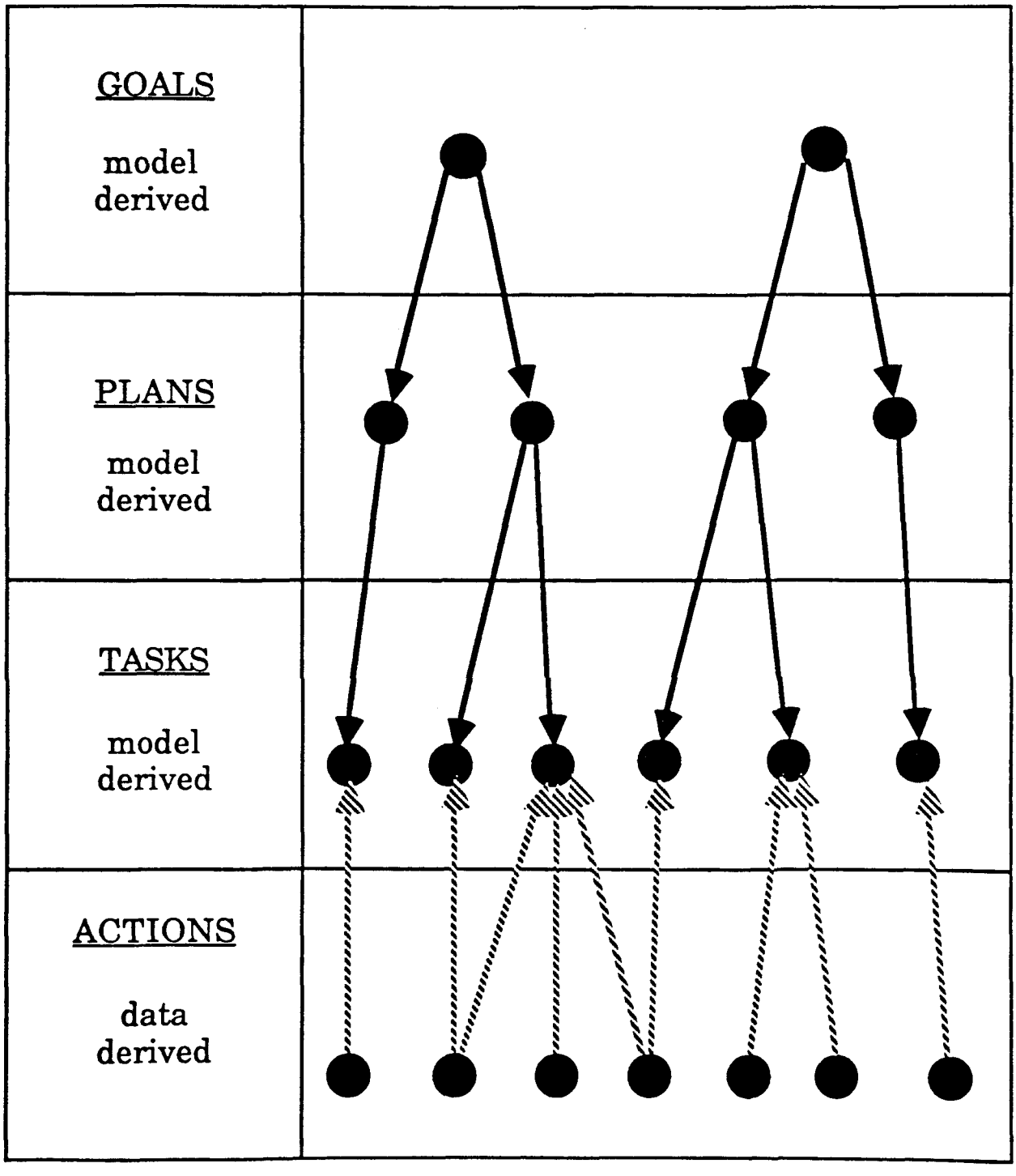


Figure 3. ACTIN's Intent Inferencing Structure

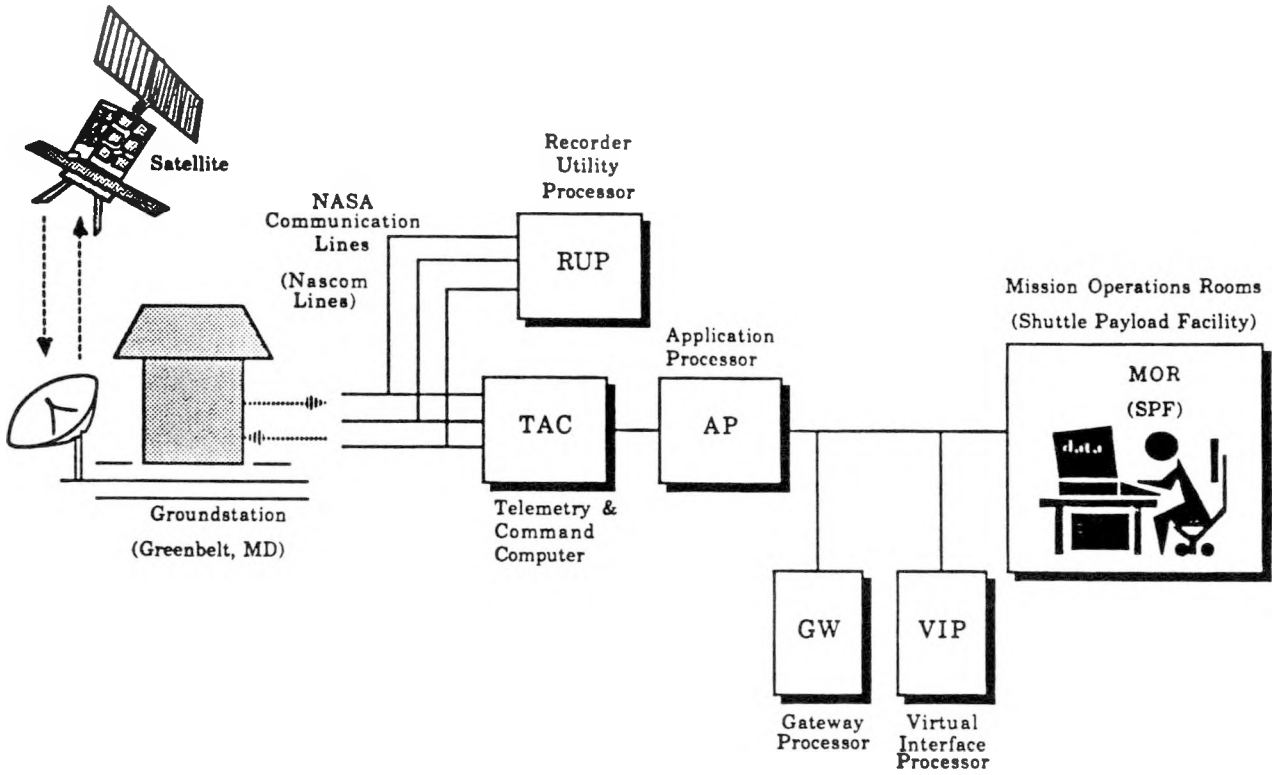


Figure 4. Multisatellite Operations Control Center

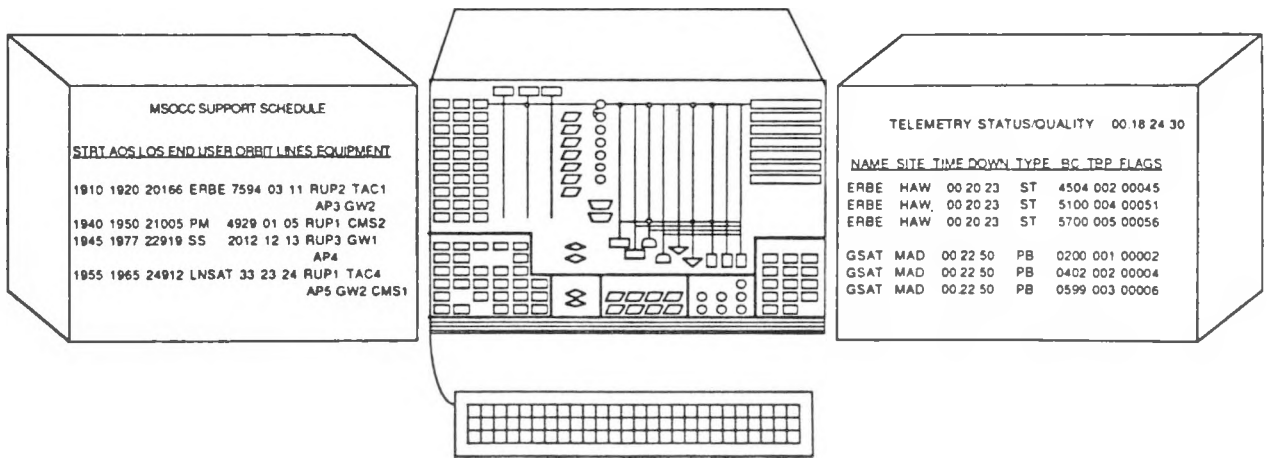


Figure 5. GT-MSOCC Workstation

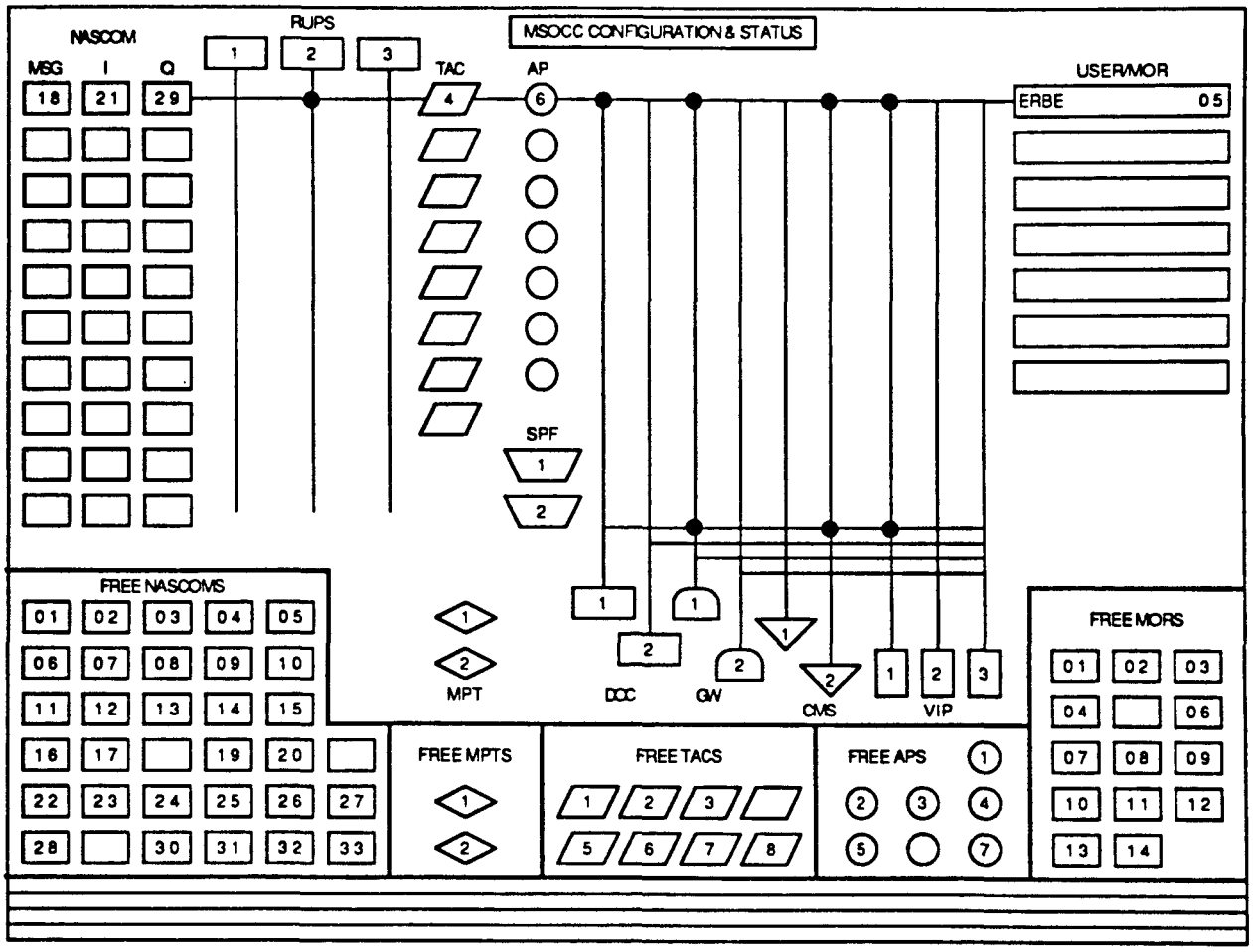
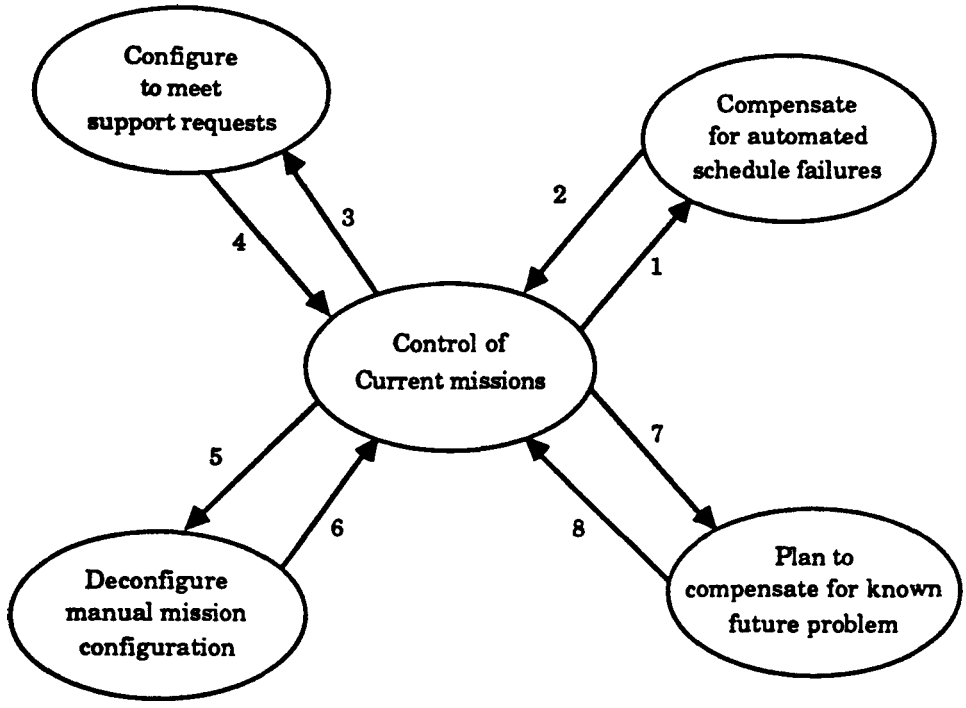


Figure 6. GT-MSOCC Center Screen



1. Error message received from the automatic scheduler.
2. Compensation completed or unable to compensate.
3. Unscheduled support request received by the operator.
4. Request configured or unable to meet the request.
5. Message received that a manually configured mission is completed.
6. Deconfiguration completed.
7. Operator summons schedule and/or mission template pages when no other triggering event takes place.
8. Terminate planning function.

Figure 7. GT-MSOCC Functions

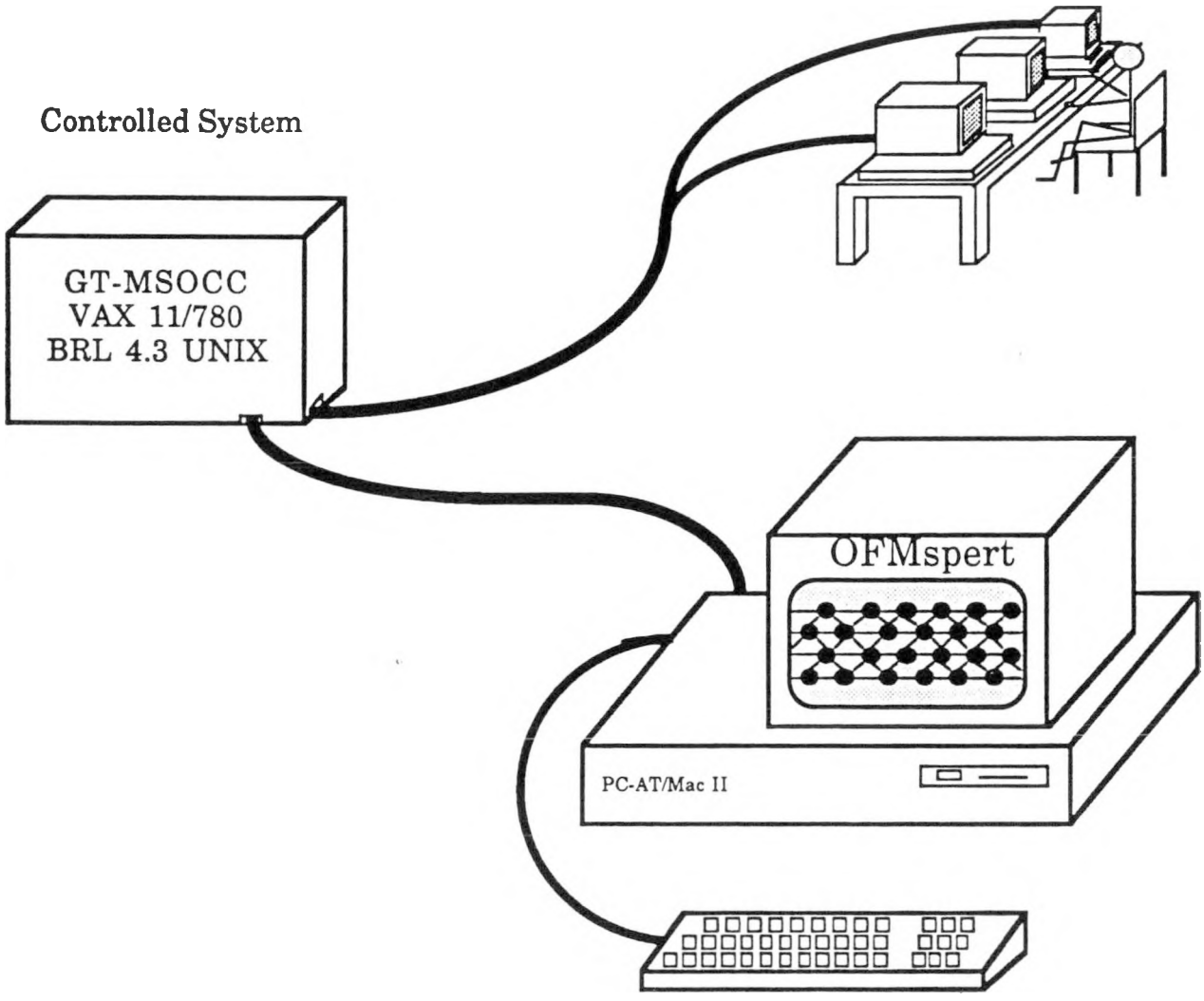
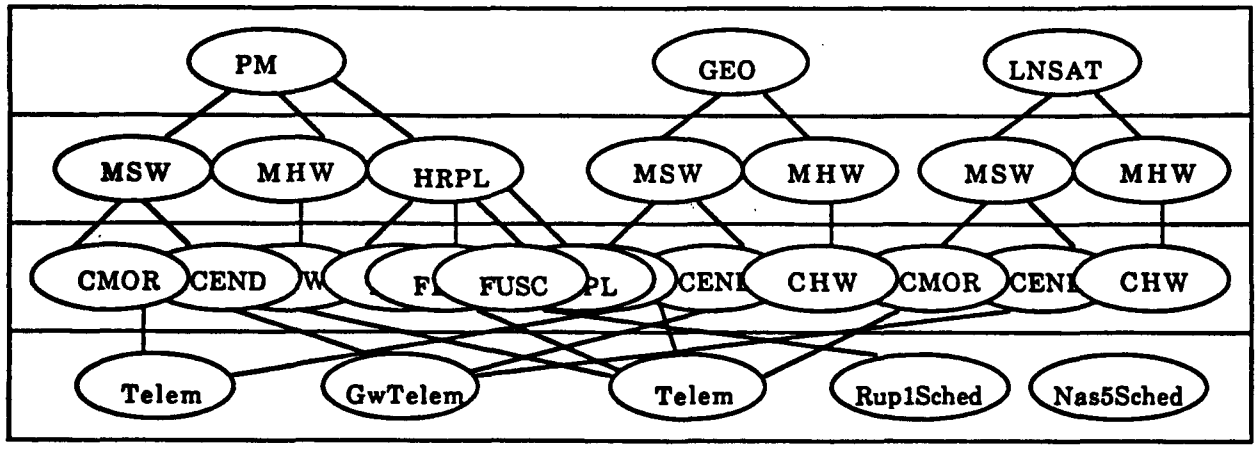


Figure 8. OFMspert-GT-MSOCC Hardware Configuration



Telem is interpreted as supporting CMOR for PM, CMOR for GEO
GwTelem is interpreted as supporting CEND for PM, CEND for GEO
Telem is interpreted as supporting CMOR for PM, CMOR for GEO,
CMOR for LNSAT, FDUR for RUP2
Rup1Sched is interpreted as supporting FUSC for RUP2
Unable to connect Nas5Sched

Figure 9. Final Blackboard

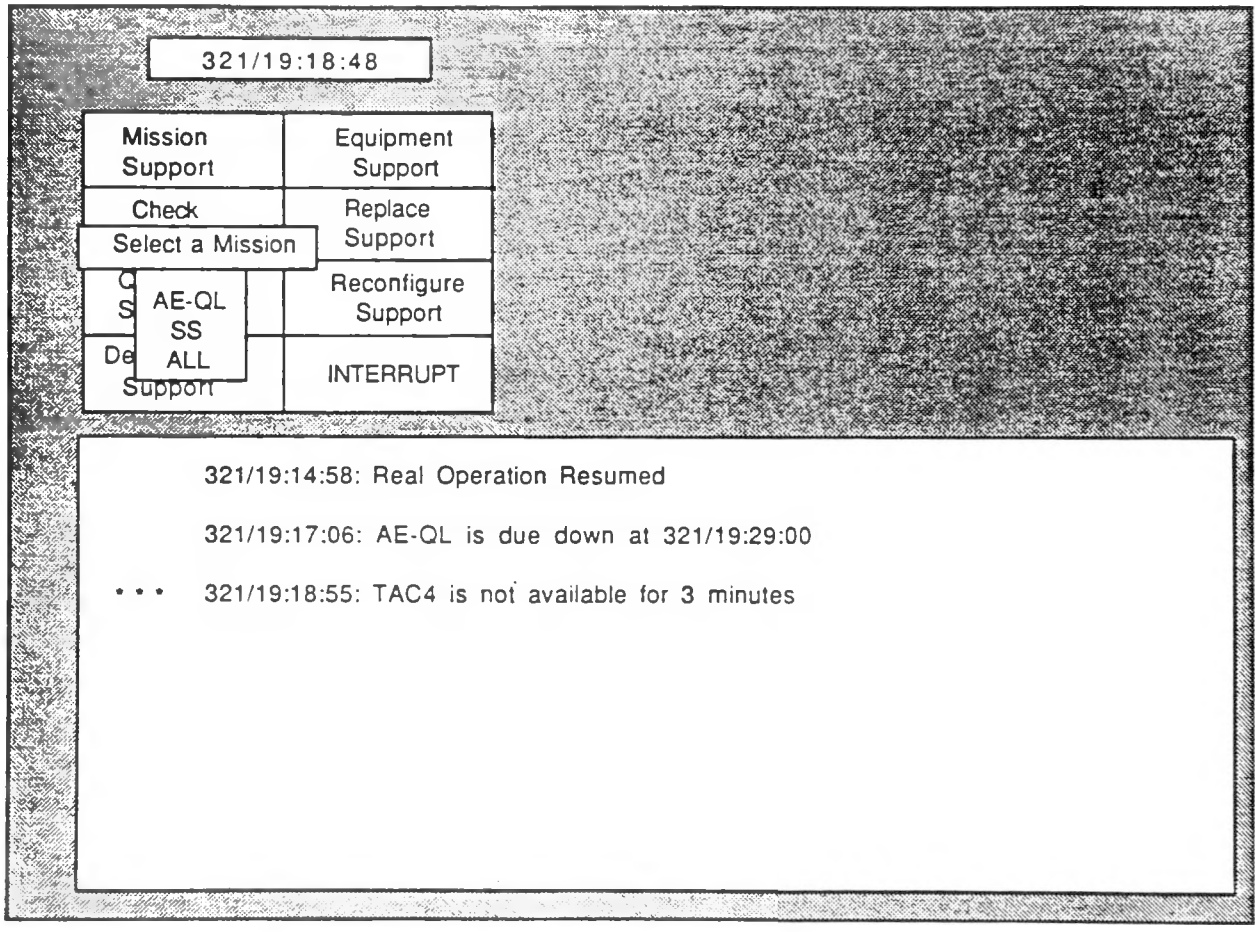


Figure 10. Example of Ally's User Interface

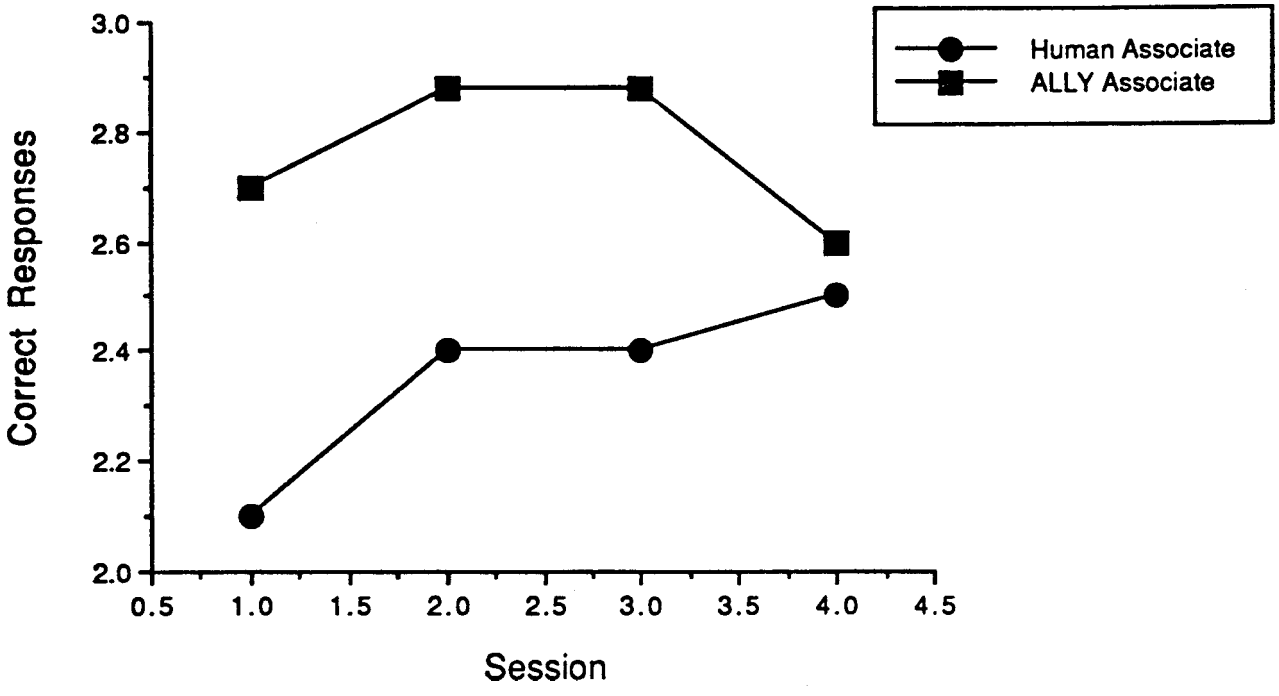


Figure 11. Mean Number of Correct Responses to Support Requests by Session

Table 1. Experiment 1: Average Percentage of Equivalent Interpretations between ACTIN and a Human Domain Expert Ordered By Rank

Configure	100%
Endpoint telemetry page requests	100
Deconfigure	97.1
Telemetry page requests	96.3
Answer	91.4
Reconfigure	91.2
Interior telemetry page requests	87.1
Replace	75.3
Mission schedule page requests	66.7
MSOCC schedule page requests	50.3
Equipment schedule page requests	21.8
Events page request	17.7
Pending page request	16.7

Table 2. Experiment 2: Proportions of Equivalent Interpretations between ACTIN and Verbal Reports

	Subject 21		Subject 22	
Telem	30/42	*	40/58	*
Endpoint Telem	33/39	*	26/38	*
Interior Telem	15/19	*	26/29	*
MSOCC Sched	36/45	*	22/31	*
Equip Sched	4/4		25/31	*
Mission Sched	8/8	*	5/7	
Events	11/18		7/15	
Pending	0/3		4/6	
Deconfig	31/31	*	30/30	*
Reconfig	7/15		6/8	
Config	5/5	*	3/3	
Replace	23/23	*	26/29	*
Answer	12/12	*	12/13	*

* Significantly good match
\# Significantly poor match

References

Bahill, A. T., Jafar, M., and Moller, R. F. 1987. Tools for extracting knowledge and validating expert systems. *Proceedings of the 1987 IEEE International Conference on Systems, Man and Cybernetics*, 2: 857–862. New York:IEEE.

Baron, S. 1984. A control theoretic approach to modeling human supervisory control of dynamic systems. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research*, 1:1–47. Greenwich, CT: JAI Press, Inc.

Bushman, J. B. 1988. Identification of an operator's associate model for cooperative supervisory control situations. Ph.D. dissertation (in progress), Center for Human–Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Chambers, A. B. and Nagel, D. C. 1985. Pilots of the future: human or computer? *Communications of the ACM* 28, No. 11: 1187–1199.

Geddes, N. D. 1985. Intent inferencing using script and plans. *Proceedings of the First Annual Aerospace Applications of Artificial Intelligence Conference*, Dayton, OH. Jones, P. M. 1988. Constructing and validating a model-based operator's associate for supervisory control,. Report No. 88-1, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Jones, P. M., Mitchell, C. M., and Rubin, K. S. 1988. Validation of intent inferencing by a model-based operator's associate, submitted for publication.

Kim, J., Gingerich, W. J., de Shazer, S., Kim, P., and I. Koh. 1987. BRIEFER: An expert system for brief therapy. *Proceedings of the 1987 IEEE International Conference on Systems, Man and Cybernetics*, 2: 853–856. New York: IEEE.

Miller, J. R., Polson, P. G., and Kintsch, W. 1984. Problems of methodology in cognitive science. In W. Kintsch, J. R. Miller, and P. G. Polson (Eds.), *Method and Tactics in Cognitive Science*, 1–18. Hillsdale, NJ: Lawrence Erlbaum Associates.

Mitchell, C. M. 1987. GT-MSOCC: A research domain for modeling human-computer interaction and aiding decision making in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, 553–570.

Mitchell, C. M. and Forren, M. G. 1987. Multimodal user input to supervisory control systems: Voice-augmented keyboard. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, 594–607.

Mitchell, C. M. and Saisi, D. L. 1987. Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. I*EEE Transactions on Systems, Man, and Cybernetics,* SMC-17, 573–593.

Nii, H. P. 1986. Blackboard systems, *AI Magazine*, 7-2 and 7-3.

Nii, H. P., Feigenbaum, E. A., Anton, J. J., and Rockmore, A. J. 1982. Signal-to-symbol transformation: HASP/SIAP case study. Heuristic Programming Project, Report No. HPP-82-6, Heuristic Programming Project, Stanford University, Stanford, CA.

Rasmussen, J. 1986). *Information Processing and Human–Machine Interaction: An Approach to Cognitive Engineering*. New York: North-Holland.

Rasmussen, J. and Goodstein, L. P. 1987. Decision support in supervisory control of high-risk industrial systems. *Automatica*, 663–671.

Rouse, W. B., Geddes, N. D., and Curry, R. E. 1987. An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems *Human-Computer Interaction*,Vol. 3, No. 2.

Rubin, K. S., Jones, P. M., and Mitchell, C. M. 1988a. OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-18, No. 4..

Rubin, K. S., Jones, P. M., Mitchell, C. M., and Goldstein, T. C. 1988b. A Smalltalk implementation of an intelligent operator's associate *Proceedings of the 1988 Conference an Object-Oriented Programming Systems, Languages, and Applications*, 234–247.

Sheridan, T. B., Charny, L., Mendel, M. B. and Roseborough, J. B. 1988. Supervisory control, mental models, and decision aids. Manuscript, Massachusetts Institute of Technology, Cambridge, MA.

Vicente, K. J. and Rasmussen, J. 1987. The cognitive architecture of decision support systems for industrial process control. Presented at the *First European Meeting on Cognitive Science Approaches to Process Control*, Marcoussis, France.

Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, CA: Morgan Kaufmann.

Wickens, C. D. 1984. *Engineering Psychology and Human Performance*. Columbus, OH: Charles Merrill.

Woods, D. D. 1986. Cognitive technologies: The design of joint human-machine cognitive system., *The AI Magazine*, 86–92.

Acknowledgements

This research was supported by NASA Goddard Space Flight Center Contract Number NAS5-28575 (Karen Moe and Walt Truszkowski, technical monitors) and by NASA Ames Research Center Grant Number HAG-413 (Dr. Everett Palmer, technical monitor). Also, this paper summarizes the intellectual endeavors of many people in the Center for Human–Machine Systems Research, particularly J. B. Bushman, P. M. Jones, and K. S. Rubin.

Blank Page

Toward A Learning Robot

Tom M. Mitchell
Matthew T. Mason
Alan D. Christiansen

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

15 January 1988

submitted to the Fifth International Conference on Machine Learning

Keywords: Learning and planning, Explanation-based methods, Empirical evaluation

Abstract

The problem of robot manipulation—of planning how to successfully grasp, push, and pull arbitrary objects to reconfigure them as desired—is an unsolved problem in robotics. We are presently developing a hand-eye system which is intended to begin with simple knowledge about how to manipulate its world, and to itself acquire increasingly refined knowledge about manipulation in specialized contexts. For example, the robot may begin with knowledge such as "to move an object somewhere to the right, push from some point on the left," and may refine the knowledge through experience into more specific, more precise assertions such as "to move object *which is in contact with a wall to the right in a straight line, push from a point on the left which is 2/3 of the distance from the wall.*" The learning method is a type of explanation-based generalization driven by an incomplete theory of the domain—in particular by a theory for qualitative differential analysis. This paper presents an approach to constructing a planning/learning system of this kind, and is primarily organized around a hand-generated example of the type of behavior our system is intended to exhibit.

---

[1]This research has been supported by NASA Ames under grant NCC 2-463, and in part by NSF under grant IRI-8740522.

Introduction

The long-term goal of the work reported here is to develop robots capable of planning and acting in worlds about which they have only an incomplete theory, and capable of automatically refining their theory of the world as a result of such attempts. As a problem in machine learning, this research can be viewed as a case study in automated theory refinement. As a problem in robotics, the motivation for such research is straightforward: Current robotic systems are successful only in very carefully engineered environments in which it is feasible for the robot to know precisely the current state of its world, and in which the robot can predict accurately the results of each action that it might choose to perform. We seek more *robust* systems which will be able to operate in less carefully engineered domains. Imagine, for example, asking a robot to "go look under that rock to see what is under it." In such a task, the robot cannot know the precise state of the world in advance (e.g., What is the size and shape of the portion of the rock which is underground?). Neither will it know with precision the consequences of actions it might perform (e.g., If it attempts to push the rock, will it budge?).

Our approach to extending the robustness of robotic systems is to allow the robot to plan relative to its approximate theory of its world and to monitor the execution of its plans. Should the robot observe that its plan is not unfolding as intended, it will seek to construct a plausible *explanation* of the possible causes of this deviation. This explanation serves as the basis for devising an error recovery strategy. It also serves as the basis for explanation-based generalization of the empirical observation, which allows refining the initial incomplete world theory.

The following sections describe in greater detail the approaches to planning with incomplete theories, plan execution monitoring, explanation-based error recovery, and automatic theory refinement. These ideas are being explored in the context of a simple hand-eye system which is described in the following section. The subsequent section describes the approach and illustrates it with a detailed, manually generated example trace. The final section summarizes the major aspects of our approach, as well as some issues raised by our initial explorations.

The Hand-Eye Testbed

Our research on learning robots is being conducted within a robot testbed based on a Puma 560 manipulator arm and IRI D256 vision system. Both the arm and the vision system are controlled by special-purpose processors that communicate with a Sun workstation running Common LISP. Common LISP functions have been developed for requesting particular arm movement, and for querying the vision system regarding the current state of the robot's world.

The problem of general manipulation is a major unsolved problem in robotics, and the

task of planning, executing, monitoring, and acquiring robot manipulation strategies is a rich area for study (see, e.g., [2], [8], [4], [5], [6], [1]). The difficulty of robot manipulation stems from both the lack of observability of the world and the complexity of the computations involved. Consider, for example, the difficulty of predicting something as simple as the exact trajectory of a block lying on the table when it is pushed by a finger from a particular point in a particular direction. The task would be burdensome, but possible if one knew the exact force applied by the finger, the coefficients of friction between the finger and block and between the block and table, the precise points of contact between the block and table, the distribution of mass within the block, etc. In fact, many of these features are unobservable in general, so the task is inherently rife with uncertainty. For these reasons, we find robot manipulation an interesting domain for studying problems of planning, error recovery, and learning under the assumption that the agent possesses only an incomplete theory of its domain.

In order to avoid difficult issues of automated perception, we have chosen a very simple world for the robot. It manipulates flat objects lying on a table, by pushing these objects. The objects may collide as they are pushed, but are never picked up off the table. This essentially restricts the robot to operating in a two-dimensional world and allows us to use simple vision algorithms to obtain reliable feedback regarding the positions, orientations, and identities of each object on the table with a delay of approximately 5 seconds. The present testbed provides a 2D world with friction (among the objects and the table), but no gravity (the objects remain in position unless pushed). We intend to alter these parameters as the project progresses, adding gravity to the 2D world by tilting the table so that objects slide to the bottom and removing friction by utilizing a table full of holes connected to an air pump so that the objects float on a cushion of air above the table. Thus, we will be able to study a variety of worlds with and without friction or gravity and study tasks such as constructing two-dimensional structures including arches, towers, and the like.

Approach

This section discusses the interrelated problems of planning, execution monitoring, error recovery, and theory refinement. Here the primary activity of the robot agent is to plan and execute actions to achieve its goals. Learning occurs in the event that planned actions fail to achieve the desired goal when they are executed. In this case, the agent attempts to explain to itself the plausible cause of the failure (in terms of a predefined theory for qualitative analysis of observed motions). This explanation is then used both for suggesting error recovery strategies and for generalizing from the observed failure to provide a more refined model of the offending action. A key assumption of the approach is that it is feasible to provide the agent a strong enough theory of its world that it can construct plausible explanations of observed failures, even though this theory may not be strong enough to predict and plan how to avoid such failures *ab initio*.

The discussion in this section presents the approach that we are presently implementing

and illustrates it in terms of a manually derived trace based on the simple planning problem shown in Figure 1. Here the robot has the goal of moving square SQ1 into the corner and begins with the following incomplete knowledge about the Push operation:

To move an object somewhere to the right
Push from somewhere on the left

The following subsections discuss how this knowledge can be used to produce a plausible plan for achieving the robot's goal and how the observed failure of the plan can lead to refining this initial knowledge to the following more specialized and more precise assertion:

To *translate square SQ1 straight* to the right *along WALL2*
Push from the left *with y coordinate of fingertip at* f_y*<-55*

Planning

Planning is the task of generating a sequence of operations, or actions, whose expected outcome achieves the desired goal. The competence of a planner in performing this task is primarily determined by the correctness of its internal models of its actions; that is, its assumptions about each action's preconditions and postconditions. If the planner's action models are insufficiently precise, the planner will be unable to produce a plan guaranteed to achieve the desired goal. The learning task we are considering here is the task of improving the precision and correctness of such action models.

More precisely, we view actions as mappings over states of the agent's world. An action model is generally a one-to-many mapping (e.g., the above model of the Push operation maps an initial world state into any of a set of possible outcome states). Following [5], we define a *strong plan* as a sequence of actions that maps the initial world state into a set of possible outcome states *each of which* satisfies the goal. Similarly, we define a *weak plan* as a sequence of actions that maps the initial world state into a set of outcome states *at least one of which* satisfies the goal.

While a detailed discussion of strategies for planning is beyond the scope of this paper, we intend for the agent to produce a strong plan when possible and to produce a weak plan when strong plans are not possible. Reference [5] describes in greater detail strategies for producing such plans.

In the present example, a strong plan is not possible, since the agent's model of the Push action is insufficiently precise. Thus, the best that can be achieved is a weak plan, such as the plan shown in Figure 1. This plan is intended to achieve the goal IN-CORNER (SQ1), but based on the agent's model of Push it can only be guaranteed to move SQ1 somewhere to the right. The plan is described in terms of the following features:

- **Goal:** the class of desired world states
- **Actions:** a sequence of actions to be executed in order to achieve the goal.

- **Expectation:** the class of possible resulting world states, based on the agent's action models
- **Intention:** the class of world states that are the intended effect of executing the planned actions. Note in the example plan the intentions are expressed as a function of the finger position f_x, which varies during execution of the Push action.

Execution Monitoring

As the plan is executed, the vision system is used to monitor the trajectory of resulting world states. Here the vision system field of view encompasses the entire world of the robot (i.e., its table), so that we avoid questions of focus of attention in perception which appear in more general robotics domains.

What is the general condition under which plan execution should be interrupted and an error signaled? Throughout the execution of the plan, the system's observations of the world state must be consistent with the plan intention. Should this cease to be the case, the plan is no longer proceeding as intended and an error should be signaled.

Figure 2 summarizes the results of executing the example plan, illustrating that the observations conflict with the plan intentions at time t_3. This corresponds to the observation that at time t_3 the square SQ1 has rotated away from the wall rather than translating directly along the wall as intended.

Error Recovery

Once an error is detected in executing the plan, one might attempt to recover simply by creating a new plan to reach the goal from the newly observed world state. However, this strategy has the drawback that the factor which led to the observed error could easily foil the new plan as well (e.g., if the block rotated away from the wall because it was being pushed from the wrong pushing point, then the new plan could easily fail for the same reason). In order to avoid this difficulty, the agent first constructs a plausible explanation for the cause of the observed error, relating the cause of the error to parameters which are within its control (e.g., the parameters to the Push operator). In constructing this explanation, the agent relies on certain general knowledge of physics which forms its background domain theory. As we will see, this derived explanation can be used to suggest a revised plan that avoids the cause of the error. Furthermore, this explanation is also used as the basis for inferring the general conditions under which the error will occur, as well as general conditions under which it can be avoided. Thus, the explanation of the cause of the error is central both to recovering from the error and to learning a general refinement *to the* agent's action models.

Explaining the Error

Let us call the difference between the plan intention and the observations the *error feature*. We will say that the system *explains the source of the error* if it answers the question "What is the dependence of the error feature on parameters under the agent's control in earlier world states?" In the present example, the error feature is the observed but unin-

tended rotation of square SQ1 away from the wall. Thus, in the present example the robot must explain how the rotation of SQ1 depends on the controllable parameters of its Push action.

Note that the explanation task here is fundamentally a problem of differential analysis: we are interested in the qualitative derivative of the error feature with respect to parameters under the agent's control. It is this explanation task that dictates the form of background domain theory required by the agent. If it can answer this question, then it can choose a new set of action parameters more tuned to bringing its observations in accord with its intentions. The agent derives the explanation by first identifying all possible influences on the error feature, then determining the type of dependence of the error feature on each influence, then determining which of these influences is under the agent's control.

Figure 3 illustrates the block-against-wall schema which contains the derived explanation of the observed error feature for the present example. The first part of this explanation determines all forces acting on SQ1 as well as the direction of the dependence of the error feature on each. The clockwise (CW) rotation depends positively on certain force components, negatively on others, and might depend either positively or negatively on another (the wall normal force) depending on the unobservable details of the contact between the wall and SQ1. This portion of the explanation follows from fairly straightforward knowledge of physics, based on the assumption that the force moment causes rotation, that forces arise from physical contacts, and that each physical contact gives rise to both normal and tangential force components. Since SQ1 participates in three physical contacts (with the table, the wall, and the robot finger), there are six force components to consider. Figure 4 summarizes the knowledge of physics used to construct this explanation.

Given this inferred dependence of the error feature on these force components, the second portion of the analysis determines the dependence of the error feature on features under the control of the robot (e.g., finger position, velocity, orientation). This is shown in the bottom of Figure 3, and follows from reasoning about the geometry of the contact between the finger and SQ1 and the dependence of the resulting forces on this geometry. This final explanation of the dependence of CW rotation on the features controllable by the robot provides the key to error recovery and to theory refinement by the robot. For instance, since the explanation indicates that CW rotation is an increasing function of the y component of the finger position, one way to decrease CW rotation is to move the finger in the negative y direction. Similarly, the rotation can also be reduced by decreasing the y component of finger velocity, or by decreasing the x component of finger velocity, but not by altering the finger orientation.

The agent attempts these candidate plan revisions in order to eliminate the undesired rotation[2]. Some of these may succeed, such as moving the finger in the negative y direction. Others may not, such as reducing the x component of the finger velocity (which reduces the rate of rotation, but not the total rotation per unit of finger travel). When candidate plan revisions are found to succeed, then the associated explanation of the error feature receives empirical validation[3], and is used as the basis for refining the agent's action model.

Theory Refinement

Once the agent recovers from its error, it uses the empirically validated explanation of the cause of the error feature as the basis for explanation-based generalization of the observed failure and later success conditions. For example, as shown in Figure 5, the explanation of the observed error feature in the present example leads to the generalization that

> Pushing SQ1 against WALL2 with *finger position f_y>-30, any theta, fv_x>10, fv_y>0* will produce CW rotation.

This generalization is supported by the observed error (observation1 in Figure 5), along with the (empirically supported) explanation of the dependence of the observed rotation on finger position, theta, etc. For example, since the explanation indicates that CW rotation is an increasing function of the y component of finger position f_y, and independent of the finger angle theta, it is reasonable to extrapolate the generalization that rotation will occur for any value of f_y greater than the value in observation1, independent of theta. Similarly, generalization2 asserts that rotation will not occur for values of f_y<-55.

Note that generalization1 is only a plausible—not a guaranteed—generalization of observation1 for the following reasons:

- The generalization is inferred by a form of extrapolation, inferring the behavior of SQ1 over some interval of pushing points based on only a single observation and the inferred partial derivation of rotation with respect to the pushing parameters at the current observed values of these parameters. There is in general no guarantee, for instance, that the analysis which produced the derivative of rotation with respect to f_y at f_y=-30 will hold over the entire interval f_y>-30, and this extrapolation is thus an approximate inference. This corresponds to an inductive bias that the derivatives of the functions of interest are slowly varying.
- The analysis of the physics is based on a number of implicit simplifying assumptions, such as the assumed uniform coefficient of friction over the surface of the table and block, the lack of jagged edges along the wall, the independence of force components and action parameters, rigid bodies, lack of inertial forces, etc. Any of these assumptions might be inappropriate, and thus the explanation on which the generalization is based might be incorrect. This corresponds to the inductive bias that the world is fairly uniform and that of all the factors influencing an object's motion, only a few terms dominate.

---

[2]In fact, the plan error may leave the world in a state such that the simple plan revision cannot be immediately applied. In this case, for example, the agent must first push SQ1 back against the wall before it can test the conjectured plan revisions.

[3]Of course the hypothesized explanation could be incorrect even if the associated error recovery strategy succeeds, since the error recovery tactic might work by coincidence or for some other reason unknown to the robot.

For both of these reasons, the robot must treat its inferred generalizations as having the same status as its initial knowledge—they are plausible statements subject to subsequent empirical disconfirmation, reanalysis, and refinement. We intend for the system to retain the explanation that justifies each proposed generalization so that this explanation can be refined as needed should the generalization be empirically disconfirmed at some subsequent time. In this light, the above inductive biases lead the agent to inductive leaps that are not firm commitments, but are rather a means of delaying consideration of additional factors until some future point at which observations may indicate a more detailed analysis is warranted.

This approach of generalizing based on plausible explanations, then subsequently elaborating the analysis on a need-driven basis, is similar to the approach proposed in [9] for dealing with explanation-based learning from intractable theories in the domain of chess. In our problem domain, we anticipate that the system's ability to automatically make and later retract simplifying assumptions to ease the analysis will be an important factor in determining the overall success of the system.

Summary and Discussion

This paper presents our preliminary insights on the problem of developing a robot system that refines its action models and therefore improves its competence with experience. We have presented an approach to automatic refinement of robot action models based on analyzing observed plan failures in terms of a predefined qualitative theory for differential analysis. This approach is presently being implemented for a robot system based on a Puma manipulator and IRI vision system. The primary characteristics of our approach are:

- Explanation-based error recovery utilizes a plausible explanation of the source of the error to hypothesize the dependence of the error feature on controllable parameters. This hypothesized dependence suggests tactics for avoiding reoccurrences of the error while recovering from the plan failure.
- Successful error recovery lends empirical support to the hypothesized explanation, which is then used for explanation-based generalization of the observed failure and success conditions.
- Learning corresponds to demand (i.e., failure) driven refinement of an initial action model, so that the initial general-but-abstract action model is incrementally refined into a hierarchy of increasingly specialized and increasingly precise models that cover past error situations.
- Learning is guided by a theory for qualitative differential analysis. This theory is itself insufficient to entail correct plans, but is very useful for analysis of errors and guiding generalization.

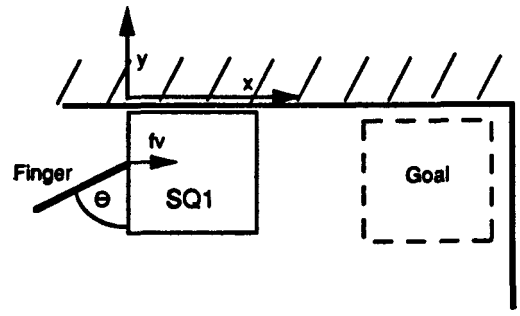
Relation to Explanation-Based Generalization

From the perspective of explanation-based learning [3] [7], this approach utilizes a particular type of incomplete theory to produce plausible generalizations of its observations. A "classic" application of explanation based generalization [7] to this problem would require a domain theory capable of explaining/proving that the observed error feature (i.e., rotation of -15 degrees) is logically entailed by the robot's Push (SQ1, 0, -30, 60, <10,0>) action. This explanation would then be used to extract just those features of observation1 which are necessary for this prediction to hold in general. But it is unrealistic to expect that the robot could produce such an explanation here, both because the physics would be too complex and because the explanation would depend on features such as coefficients of friction, precise contact characteristics, etc., which are *unobservable* by the vision system.

Given this difficulty in applying straightforward explanation based generalization, our approach is to instead rely on a qualitative theory for differential analysis of motions, which is less complex and does not depend on knowing precise values of physical parameters such as coefficients of friction. It is used to explain the *sign of the difference between observed and intended values of world state variables* rather than the precise observed values. This explanation is then used to suggest *directions for changing the initial action parameters* so that this error feature will be reduced. Should the suggested changes be tested and found to have the predicted effect, then the corresponding explanation receives empirical support and is used to drive the agent's generalization process. This process extrapolates from the specific training instance to a general class of situations and action parameter values for which the action effects can be more precisely predicted. It is interesting that this theory for differential analysis is very helpful in guiding learning despite that fact that it is not useful for initial planning (e.g., it is not useful in selecting the parameters for the Push action in the initial plan). See [10] for a discussion of methods for qualitative differential analysis.

A number of questions for further research are raised by this approach: Will the generalization errors introduced by the approximate theory and the extrapolation mechanisms overwhelm the system, or will it be able to incrementally refine its beliefs as it discovers these errors during subsequent activities? Can this type of differential analysis be useful for analyzing plan execution errors that are not typically described numerically (e.g., the tower was intended to remain intact, but it fell down)? Can this differential analysis be used to automatically design feedback control loops that use the inferred dependence of the error feature on controllable parameters to continuously update the control parameters of the robot's action? Can the approach be extended to deal with situations in which the cause of the plan failure is outside the scope of the background theory (e.g., if the finger is magnetic)?

The Problem:



Where:

coordinate frame origin: x=0, y=0 is initial position of top left corner of SQ1
position of SQ1 described by <x,y,theta>, where
x = x coordinate of top left corner of SQ1
y = y coordinate of top left corner of SQ1
theta = orientation relative to initial orientation of SQ1

f_x = x coordinate of tip of finger
f_y = y coordinate of tip of finger
f_{theta} = orientation of finger (independent of finger velocity)
fv = the vector velocity of the finger tip (independent of f_{theta})
$SQ1_{x\text{-initial}}$ = x coordinate of initial position of SQ1

Initial Knowledge:

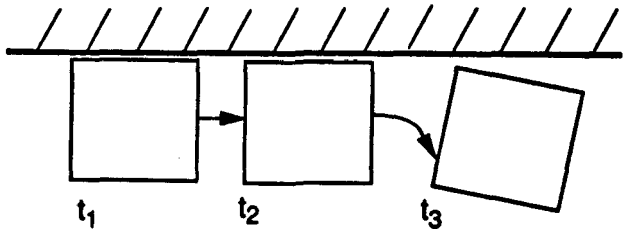
To move ?object somewhere to the right
Push(?object,?f_x,?f_y,?f_{theta},?fv) with ?f_x=x coordinate of left edge of ?object

The Plan:

Initial State:
AT(SQ1, <0,0,0>)
Goal: (target world state)
AT(SQ1, <200,0,0>) (i.e., IN-CORNER(SQ1))
Actions: (sequence of planned actions)
Push(SQ1, 0, -30, 60, <10, 0>)
Expectation: (predicted trajectory of world states)
AT(SQ1, <x,y,theta>), where x > $SQ1_{x\text{-initial}}$, y<0, 0<theta<360
Intention: (target trajectory of world states)
AT(SQ1, <x,0,0>), where x=f_x

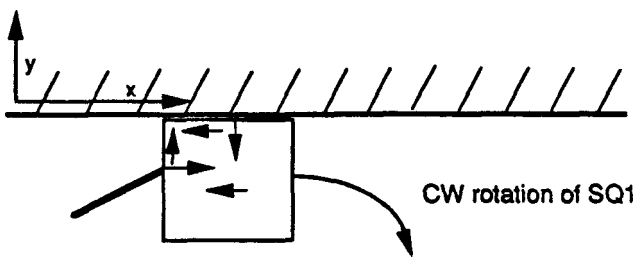
Figure 1. Problem and Plan to Achieve IN-CORNER(SQ1)

Expectation: (predicted trajectory of world states)
AT(SQ1, <x,y,theta>), where $x > SQ1_{x\text{-initial}}$, y<0, 0<theta<360
Intention: (target trajectory of world states)
AT(SQ1, <x,0,0>), where $x=f_x$
Observations: (observed trajectory of world states)
t_1: AT(SQ1, <0,0,0>)
t_2: AT(SQ1, <3,0,0>)
t_3: AT(SQ1, <6,-1,-15>) **



** Error: At t3, difference between intended and observed position of SQ1: <0, -1, -15>
(i.e., intended translation (y and theta should remain 0), but observed rotation)

Figure 2. Execution Monitoring



Note arrows inside SQ1 represent force components.

Applicability conditions:
IS(Block ?b)
IS(Wall ?w)
IN-CONTACT(?w ?b)

Force components acting on block	Dependence of CW rotation on force
Wall normal	? (+ or -)
Wall friction	-
Table normal	0
Table friction	-
Finger normal	+
Finger friction	+

Controllable features	Dependence of CW rotation on feature
Finger $position_y$ (i.e., $?f_y$)	+
Finger $velocity_y$ (i.e., $?fv_y$)	+
Finger $velocity_x$ (i.e., $?fv_x$)	+
Finger orientation (i.e., $?f_{theta}$)	0

Figure 3. Block-against-wall Schema

- Forces arise from physical contact
- Contact forces can be characterized in terms of two components:
  - Normal component (normal to contact surface)
  - Tangential component (due to friction)
- Friction:
  - For translation, friction force directly opposes direction of translation
  - For rotation, friction resists direction of rotation
- Motion = Translation + Rotation
- Translation depends monotonically on vector sum of forces
- Rotation depends monotonically on moment of forces

Figure 4. Primitive Qualitative Physics

- Observation1: CW rotation for SQ1 against WALL2 with finger at position f_y=-30, theta=60, ...
- Explanation: Sum of forces from wall, table, finger produce positive torque. CW rotation is an increasing function of finger $position_y$, finger $velocity_y$, finger $velocity_x$.
- Candidate Plan Revision: Move finger $position_y$ in negative y direction to reduce rotation.
- Observation2: Pure translation (no rotation) when pushing at position f_y=-55.
- Generalization1: Pushing SQ1 against WALL2, with finger position f_y>-30, *any theta*, fv_x>10, fv_y>0 ...., will produce CW rotation.
- Generalization2: Pushing SQ1 against WALL2, with finger at position f_y<-55, *any theta*, fv_x<10, fv_y<0, ..., will produce no CW rotation.

Figure 5. Summary: Observation, Explanation and Generalization

References

[1] Brost, R. C. Automatic Grasp Planning in the Presence of Uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, San Francisco, 1986.

[2] Carbonell, J. C., and Gil, Y. Learning by Experimentation. In Langley, P. (editors), *Proceedings of the Fourth International Workshop on Machine Learning*, pages 256-266. Morgan-Kaufmann, Irvine, June, 1987.

[3] DeJong, G., and Mooney, R. Explanation-Based Learning: An Alternative View. *Machine Learning* 1(2):145-176, 1986.

[4] Donald, B. R. *Error Detection and Recovery for Robot Motion Planning with Uncertainty*. Technical Report 982, MIT-AI, July, 1987.

[5] Lozano-Perez, T., Mason, M. T., and Taylor, R. H. Automatic Synthesis of Fine-Motion Strategies for Robots. *International Journal of Robotics Research* 3(1):3-24, 1984.

[6] Mason, M. T. Mechanics and Planning of Manipulator Pushing Operations. *International Journal of Robotics Research* 5(1), 1986.

[7] Mitchell, T. M., Keller, R. K., and Kedar-Cabelli, S. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1(1), 1986.

[8] Segre, A. M. *Explanation-Based Learning of Generalized Robot Assembly Plans*. PhD thesis, Univ. of Illinois, 1987.

[9] Tadepalli, P. Learning Approximate Plans in Games. December, 1986. Rutgers Computer Science Ph.D. thesis proposal.

[10] Weld, D. Comparative Analysis. *Artificial Intelligence:* to appear, August, 1988.

Blank Page

Section 5: Breakout Session Summaries

Blank Page

Breakout Session: Human-Machine Communication

December 5, 1988

Marty Beckerman
Oak Ridge National Laboratory

Attendees:

Bill Knee — Oak Ridge National Laboratory — Chair
Marty Beckerman — Oak Ridge National Laboratory — Reporter
Prem Chopra — University Tennessee at Chattanooga
Harold P. Van Cott — National Academy of Sciences
James H. Graham — University of Louisville
Thomas Hutchison — University of Virginia
Kazuhiko Kawamura — Vanderbilt University
Christine Mitchell — Georgia Institute of Technology
Amit Mukerjee — Texas A&M University
Michael J. Rabins — Texas A&M University
Karl Reid — Oklahoma State University
Thomas Sheridan — Massachusetts Institute of Technology
Phil Spelt — Oak Ridge National Laboratory

Question: What are the most effective means of communication between man and machine in cooperative control systems involving physical processes?

The participants' interests in human–machine symbiotic systems were varied, ranging from the design of systems for handicapped individuals such as quadriplegics (Prem Chopra) to the building of near-earth satellite operator's associates for NASA (Christine Mitchell) to robotics for handling nuclear power plant accidents. At the conclusion of the session, the group on Human–Machine Communication observed that there are two immediate, promising areas of research into communications in human–machine symbiotic systems. These high priority areas are: (1) The simultaneous (concurrent) and integrated use of multiple modalities, or sensor channels, for communication between man and machine and (2) the development of models of the human operator within intelligent computer systems, with emphasis on characteristics present during periods of both nominal and unusual arousal and stress.

The agenda for the Human–Machine Communication group as framed by Bill Knee included the identification/discussion of relevant issues, the formulation of an answer to the representative question of what is the most effective means of communication between man and machine in cooperative systems, and the prioritization of issues and the associated research. To provide a framework for discussion, the participants examined the schematic diagram of a cooperative system presented by Tom Sheridan in his keynote talk (Figure 1) and a similar, but more detailed and therefore less general, plot by Bill Knee (see below). These discussions were intended to serve as a first pass as to the general properties of cooperative systems regarding communications. This initial group discussion was followed by one in which the participants examined symbiotic systems with respect to their human, task, and environment characteristics.

Among the various task characteristics were time available, rate of information flow, resource requirements, fault (error and uncertainty) tolerance, closed/open environment, level of impact (cost of task), and risk. Among the environment parameters of importance were those relating to cultural and regulatory differences and those of a physical nature which produce, say, noise and signal degradation and hostility (high temperature, radiation, vibration, smoke, etc.).

The systems of interest to the participants involve a broad range of tasks and environments. However, when viewed with respect to their human characteristics and communication/interface requirements, the symbiotic systems become remarkably similar. This observation became the focus for the continuing discussion. The human has multiple information processing channels, but most, if not all, systems do not take advantage of this characteristic. The human has a need for integrated, pictorial information, has problems dealing with stress, inattention (boredom), and the rapid transition from situations where either immediate or no action is required and can suffer from cognitive overloading.

A paradigm for human–machine symbiotic systems may be of a complex, high-technology environment in which at least partially unknown situations arise and in which performance under stress is required of the human. The importance of building human confidence in diagnoses provided by the computer/machine subsystem in emergency situations was noted. The events which took place at Three Mile Island and the Vincennes affair were discussed within the framework of the emerging paradigm.

The two recommendations stated at the beginning of the summary address these issues of paramount importance. The first recommendation is motivated by the need to make better, if not full, use of man's sensory capabilities. Regarding the second recommendation, models of the human operator within the machine allow it to form expectations related to its human counterpart. As such, the machine can "act" more intelligently and, therefore, can aid in building a situation that supports trust and understanding. In addition, models of the human permit the machine to infer intent on the part of the operator, thereby enhancing the environment for understanding. Underlying this is the observation that in truly symbiotic systems, good performance is supported by knowing what your partner/associate is or is not capable of.

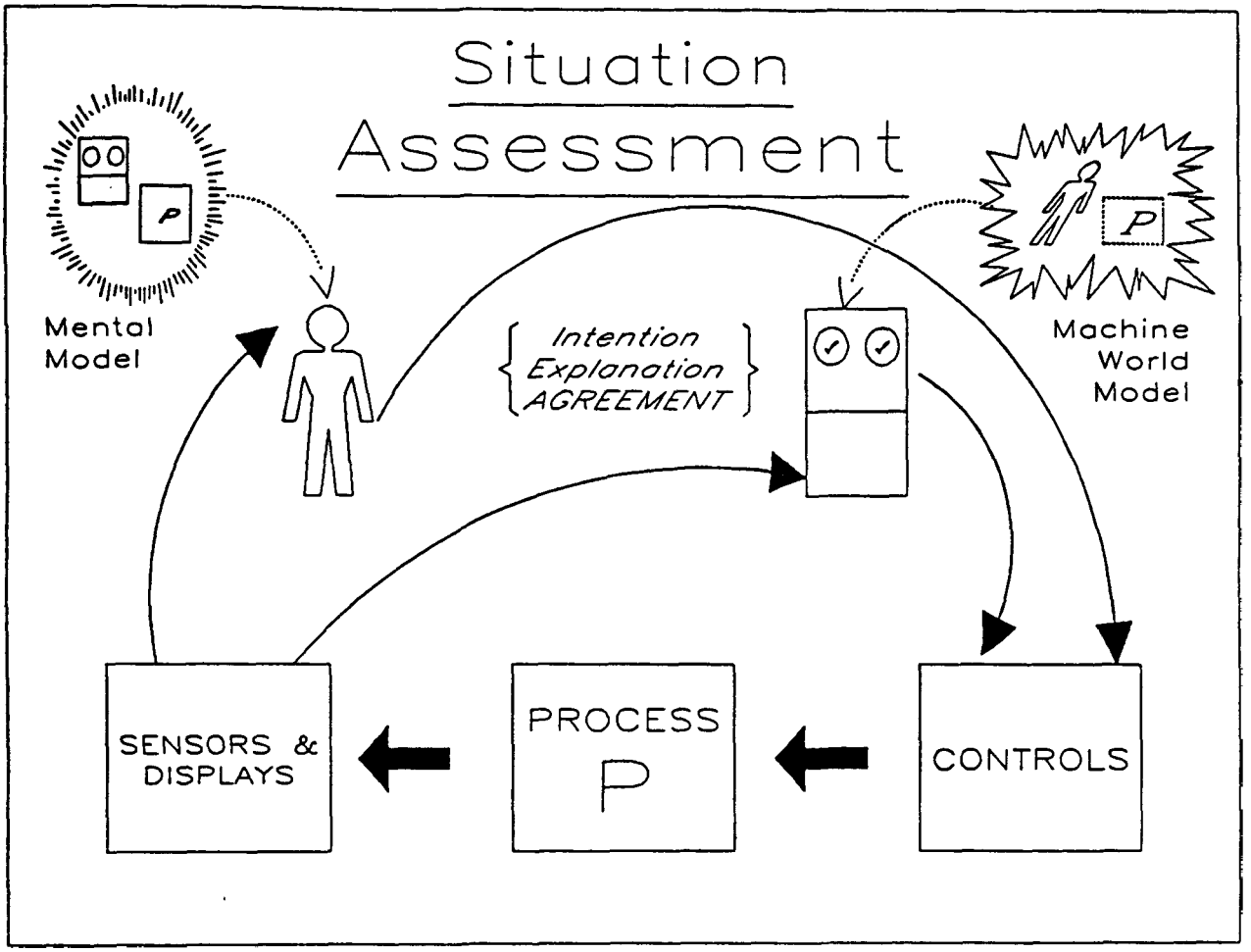


Figure 1. Schematic Diagram of a Cooperative System

Blank Page

Breakout Session: Autonomous Task Planning and Execution Monitoring

December 5, 1988

Reinhold C. Mann
Oak Ridge National Laboratory

Attendees:

Gerard DeSaussure — Oak Ridge National Laboratory — Chair
Reinhold C. Mann — Oak Ridge National Laboratory — Reporter
Kai-Hsiung Chang — Auburn University
Fred DePiero — Oak Ridge National Laboratory
Maria Gini — University of Minnesota
Claudio Gutierrez — University of Delaware
Susan Hruska — Jacksonville State University
Avi Kak — Purdue University
Francois Pin— Oak Ridge National Laboratory
Ching-Long Shih — University of Kentucky
James Tulenko — University of Florida
Yuan Zheng — Clemson University

Question: What are the most promising approaches to real-time task planning and execution monitoring between heterogeneous resources, at least one of which is human?

Summary

The discussion in this session focused on mainly two areas: (1) interpretations of the scope of and rationale for human–machine symbiosis and (2) issues involved in autonomous task planning.

Figures 1 and 2 summarize most of the arguments made during the first part of the discussions.

Symbiosis can be viewed as an approach to bridge the gap between the majority of robotic systems today that are capable of performing complex tasks under full teleoperation or after being preprogrammed and those few prototype systems capable of autonomous behavior for simple assignments.

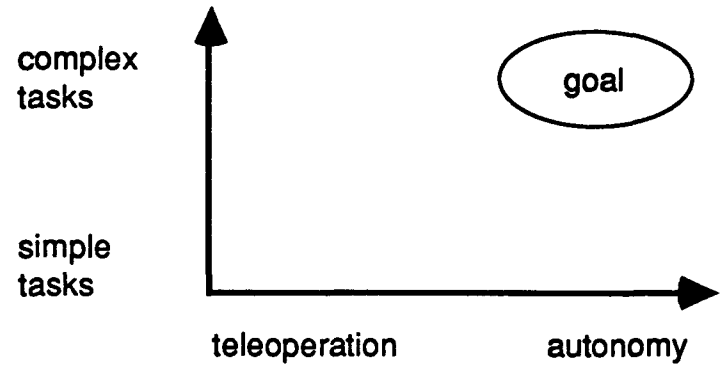


Figure 1

The issue in symbiosis is not autonomy versus teleoperation but a dynamic mixture of both modes of robot operation. It was pointed out that the concepts considered in symbiosis do not apply only to human–robot systems. An example presented was the interaction between a human and an intelligent data base system through a computer work station. In any event, autonomy is a prerequisite for symbiosis. Furthermore, the group maintained that autonomy is not synonymous to intelligence.

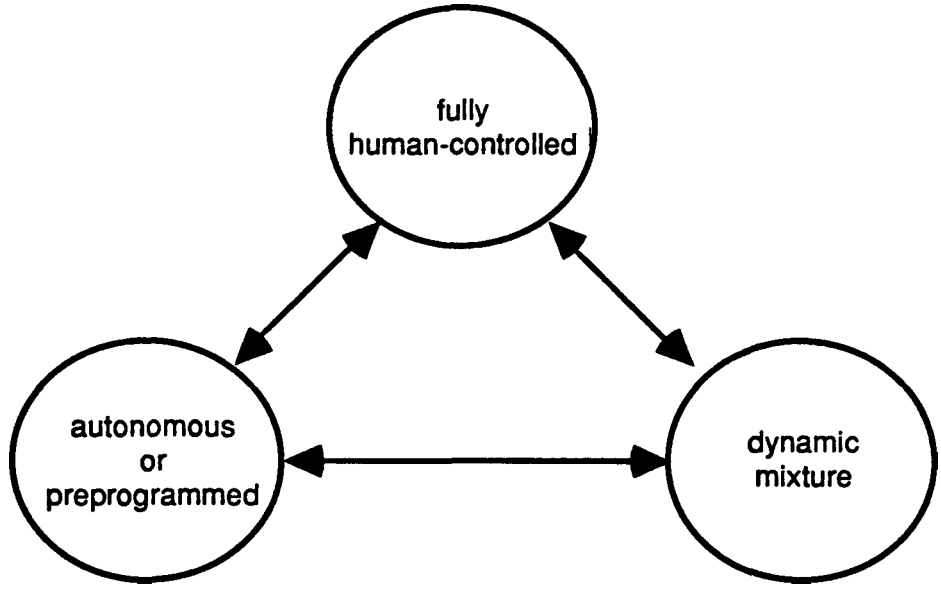


Figure 2

Task planning refers to the decomposition of a symbolic goal description into a sequence of tasks with associated numeric descriptions that can be executed by the different agents in the symbiotic system. Prerequisites for task planning include an appropriate goal description; models for the current environment and agents and other resources available to the system, as well as adequate definitions of possible constraints under which the goal must be achieved, e.g., limited time.

There was consensus among the participants that it is difficult if not impossible to separate task planning from task allocation and that the issue of learning is of great importance in making the transition from task planning entirely based on human decision making to autonomous planning by an intelligent machine system. Since the ability to generalize is crucial for learning, current connectionist approaches were considered to be less promising than cognitive approaches. Task planning for a symbiotic system should take advantage of superior human capabilities in top-level planning and good planning abilities of existing automated systems at the low level and should allow for human intervention required for producing plans for mid-level tasks and unforeseen situations as well as learning from these interventions.

The group concluded that among the main open issues in autonomous task planning are adequate modeling of agents and resources in the symbiosis system, learning leading to the capability to generalize, and the smooth transition between human intervention and autonomous planning at different task levels.

Blank Page

Breakout Session: Dynamic Task Allocation

December 5, 1988

Wayne Manges
Oak Ridge National Laboratory

Attendees:

Lynne Parker — Oak Ridge National Laboratory — Chair
Wayne Manges — Oak Ridge National Laboratory — Reporter
Kevin Corker — BBN Laboratories
Bill Hamel — Oak Ridge National Laboratory
Jia-Yuan Han — Southern Illinois University
Scott Harmon — Robot Intelligence International
Chuck Jorgensen — Thomson-CSF
Thomas Mitchell — Carnegie-Mellon University
Eui Park — North Carolina A&T State University

Question: What are the best methods of allocating cooperative human-machine tasks?

Abstract

The group began by developing a list of the attributes of the "best" approaches to dynamic task allocation. The resulting list included such factors as timeliness, coherence, load balance, and risk. Subsequent discussion detailed the importance of various architectural, human interface, and measurement issues in designing a human-machine symbiotic system that optimizes the allocation attributes for the current application. The group concluded by emphasizing the importance of on-line "observers" and other intelligent mechanisms to dynamically assess the progress and directedness of the agents of the dynamic task allocation implementation in achieving the defined tasks. Topics recommended for further research included improved measurements, architectures, relative human–machine roles, and task granularity.

Assumptions

To begin addressing the initial question posed to the breakout session, the group agreed that some basic assumptions would be useful to help bound subsequent discussions. It was agreed that dynamic task allocation is a desirable goal since static allocation, though cheaper to implement in terms of design costs, implementation costs, and run-time communication and computing costs, cannot be effectively used in dynamic environments and is usually less reliable. The group agreed that symbiotic teams did not necessarily have to consist strictly of one human and one machine; instead, we chose to consider the concept of a human master working with several automated apprentices as well as the multiple human, multiple machine cooperative relationship. By assuming that the agents are willing to cooperate, resource contention problems can be resolved by invoking global goals. The group also wanted to include discussions concerning the computer-controlled human by addressing the psychological, physiological, and interface implications of a human allowing a computer to assume control under certain conditions. The final assumption recognized the fact that symbiosis is a realistic goal, since examples of elements of the human-machine symbiosis concept currently exist. The pilot's associate, the Mars rover, and applications in areas of hazardous waste handling are examples where symbiotic systems are being developed or considered.

Discussion

The group began considering the question of the best methods of dynamic task allocation by defining the attributes present in an optimal, or at least "good," allocation strategy. Although the relative importance among the allocation attributes depends on the application, improvements in the attributes for a particular application would indicate an improvement in the allocation strategy. The attributes considered important in human-machine symbiotic systems include timeliness, coherence, load balance, and risk. Timeliness would be the overriding attribute in task allocation strategies for time-limited applications. In such applications, the allocation strategy would assign tasks by assessing the speed with which certain tasks are performed by the available agents. The allocation procedure itself may be restricted in these situations due to a limit on the time available for an allocation decision to be made. Coherence is a crucial factor for most, if not all, applications and requires that the focus of subtasks remain directed toward a high-level, common goal. The load balance attribute is predominate in applications requiring the maintenance of a critical level of activity in all agents of the symbiont system. Finally, the risk minimization attribute involves consideration of the probability of jeopardizing mission success or the symbiont system itself in selecting a suitable task allocation. Further discussion noted that it is essential that a task allocation strategy be adapted in real time in response to an environmental change, an internal fault detection, or a shift in goals. In response to certain emergencies, timeliness may become the most important attribute.

The remainder of the dynamic task allocation discussion focused on concerns of measurement, human factors, architectures, and future research. The discussions of measurement emphasized the need for dynamic assessment of the "goodness" of the current task allocation strategy to allow the improvements in subsequent strategies to be verified. Measurement techniques also encompass the need for evaluating the capabilities of the human(s) and the machine(s). One area where measurement techniques already exist that might assist in dynamic task allocation is in the taxonomy of learning. For a number of years, research in this field has centered on assessing the capabilities of humans. Much work has been accomplished in evaluating human learning in skills, knowledge, abilities, and attitudes (SKAAs). Appropriate metrics may already exist in these areas that could be adapted to symbiotic systems. In discussions concerning related techniques for measuring machine performance, some group members believed that machine capabilities are best assessed by examining the hardware and software that make up the machine rather than external attributes. Whether examining human or machine capabilities, it was recognized that the agent capabilities may change over time. To measure the relative performance of the agents as well as improvements over time by a particular agent, benchmarking (using a repeatable task for which a good metric is developed) was suggested as a potential solution.

The human factors issues centered on the roles the human(s) and machine(s) should play both in the allocation of tasks and in the execution of tasks. Of particular interest were questions concerning the consequences of permitting a human to be controlled by a machine. Additional issues included the need to recognize human stress and attitude in the selection of an appropriate task allocation.

The architectural issues focused on performance, available communication bandwidth, and potential flexibility for adjusting the allocation strategy. These discussions were triggered by a recommendation that the task allocator and the execution monitor should be considered elements of a closed-loop control system. Discussions of committee versus hierarchical architectures concentrated on their relative advantages and disadvantages in poorly- and well-understood environments. The importance of understanding the tasks to be performed was considered crucial to establishing the appropriate task allocation architecture. Heteroarchical (combination of hierarchical and committee) architectures were considered most applicable to symbiotic systems.

Suggested Research Topics

In addition to the discussion of the issues detailed above, the group identified some areas of human–machine symbiosis that were considered important for advancing the state of the art in dynamic task allocation for the next few years:

1. Measurements—Good benchmarks and techniques for quantifying and measuring human and machine skills/capabilities are needed. Incorporation of results from research in human learning may be important here.

2. Architectures—The relationship between execution monitoring and task allocation influence architectures. Would dynamic architectures be advisable so that hierarchies could develop inside the committee structures? Are dynamic architectures even possible or desirable?

3. Roles—What are the psychological and physiological impacts of computer-controlled humans? When is it important for humans to assume control even if they are not the "best" choice? What data from the human are important in assessing the relative roles?

4. Task granularity—What task granularity assures the smooth transition among elemental subtasks as they are assigned to the different agents?

Conclusions

The group agreed on a list of attributes that can be used to assess the goodness of a dynamic task allocation strategy and determined that the relative weights of the attributes are context specific. We also agreed on the importance of both off-line skill measurement and on-line assessment of human or machine task execution performance as feedback to the dynamic task allocation algorithm. Finally, the group noted that the coupling among the modules in the dynamic task allocation strategy requires high performance, highly flexible architectures. We concluded that the best dynamic task allocation strategies are those that maintain metrics for real-time assessment and adjustment of the strategy in response to changes in the environment, modifications of the goals, or variations within or among the agents.

Breakout Session: Human-Machine System Architecture

December 6, 1988

Fred W. DePiero
Oak Ridge National Laboratory

Attendees:

William R. Hamel — Oak Ridge National Laboratory — Chair
Fred W. DePiero — Oak Ridge National Laboratory — Reporter
Marty Beckerman — Oak Ridge National Labortary
Prem Chopra — University of Tennessee at Chattanooga
Kevin Corker — BBN Labs
James H. Graham — University of Louisana
Jia-Yuan Han — Southern Illinois University
Thomas Hutchison — University of Virginia
Kazuhiko Kawamura — Vanderbilt University
Wayne Manges — Oak Ridge National Laboratory
Christine Mitchell — Georgia Institute of Technology
Amit Mukerjee — University of Michigan
Thomas P. Sheridan — Massachusetts Insitute of Technology
James S. Tulenko — University of Florida
Yuan F. Zheng — Clemson University

Question: What human–machine system architectures allow real-time cooperative interaction between the human and the machine?

Abstract

This session of the workshop focused on the subject of human–machine symbiotic (H–MS) system architecture. The session began with an agreement on the definition of an architecture. Two examples were discussed, and then, to focus the discussion, the following question was considered: "Is there a canonical human-machine symbiotic system architecture, or are there only good design rules?" This question provoked many suggestions of what a "good" architecture might be. It became clear that a set of criteria was needed to evaluate the merits of an architecture. Throughout the session many areas of potential research were also identified.

Definition of an Architecture

The possibility of working with either a hardware or a software type of architecture was discussed. It was decided that a functional type was more suited to describing a system's objectives. It was agreed that the hardware and software architectures together give a system its functionality, and hence would be an equivalent representation of a system. However, the functional description was more removed from implementation and performance related details and was thought to be a more appropriate basis for the discussion.

It was suggested that a system designer's notion of an architecture can be very different from a user's impression of the same system. The session decided to work with this idea. An 'architecture' was then taken to mean: "A functional description of a system's structure, as viewed from a user's perspective." It was noted that this expands the concept of a human–machine interface (HMI) from being just a port to being more of a viewpoint.

Example Architectures

Two members of the session presented their own architectures to the group. The diagrams are included here for the reader's consideration, but an attempt will not be made to fully explain each. The first is an architecture associated with a particular project and is shown in Figure 1. The second, shown in Figure 2, is not specific to a project but has been used as more of a guideline for projects. Its components describe where development efforts are often concentrated. Because this second architecture was more general purpose (it was not motivated by the objectives for a particular system), there was a general agreement that a representation of this style approached more of a canonical form. A notable difference in the two examples is the connection of the HMI to the rest of the system. The first puts the HMI into one module. The second does not attempt to encapsulate it; instead, the HMI reaches all layers in the architecture (except for the low level robot control).

If one generalizes a system architecture as hierarchical layers, it is clear that the layers will have a top-down structure. In addition, the most general case may be a heteroarchical architecture, containing "committee" structures within the more structured layers. The corresponding architecture of the HMI will correlate with this overall system structure as a function of two basic features: 1) the specific human's functional responsibility and 2) the specific HMI requirements for a particular architectural layer.

Criteria/Wish List for a Good Architecture

Many qualities of architectures were discussed at this point in the session. A number of these were very legitimate metrics for systems in general but were not qualities that applied to H–MS architectures in particular. These were qualities like modularity, data accessibility, fault tolerance, and others. An important criterion that did apply to H–MS systems was the role of the human-machine interface. By definition, an HMI must exist in some form in any human–machine symbiotic system. There was an agreement that the specific elements and

connectivity of an HMI are dependent on its functional requirements. (This supported the idea of a very general canonical form.) For example, note again the very different connectivity of the HMI in the examples given. There was a disagreement as to whether or not a high degree of connectivity between the HMI and all other layers of a hierarchy was appropriate. Some felt it should only connect to a subset of the layers; others thought there should be no restrictions. The group was then reminded of its adopted definition of an architecture. There was an agreement that an HMI should not be restricted from any possible connections, provided that a particular link is appropriate for a particular user. This idea spawned the concept of a dynamic architecture that can change to meet the needs of the user or the needs of the system.

Another aspect of an H–MS system that can be affected by its architecture is the aggregate stability of the system. More specifically, this is the problem of maintaining stability while some parts of a system are being driven directly by human control and other parts run in an autonomous mode. This problem in hybrid human–machine control mixture was thought to be an interesting research topic as it applies to system architecture design.

As the session continued to define quality measurements of an H–MS architecture, the subject of developing these metrics themselves was noted as an interesting research topic. Examples of good techniques with poor applications were mentioned. This motivated the idea that metrics which compare effectiveness vs. complexity would be worthwhile.

Summary

The session members felt that if a canonical form of an H–MS system does exist, then it would at least have the two following properties. First, it would be a high level template only. Secondly, the perspective that the human interface provides should vary according to the user's needs.

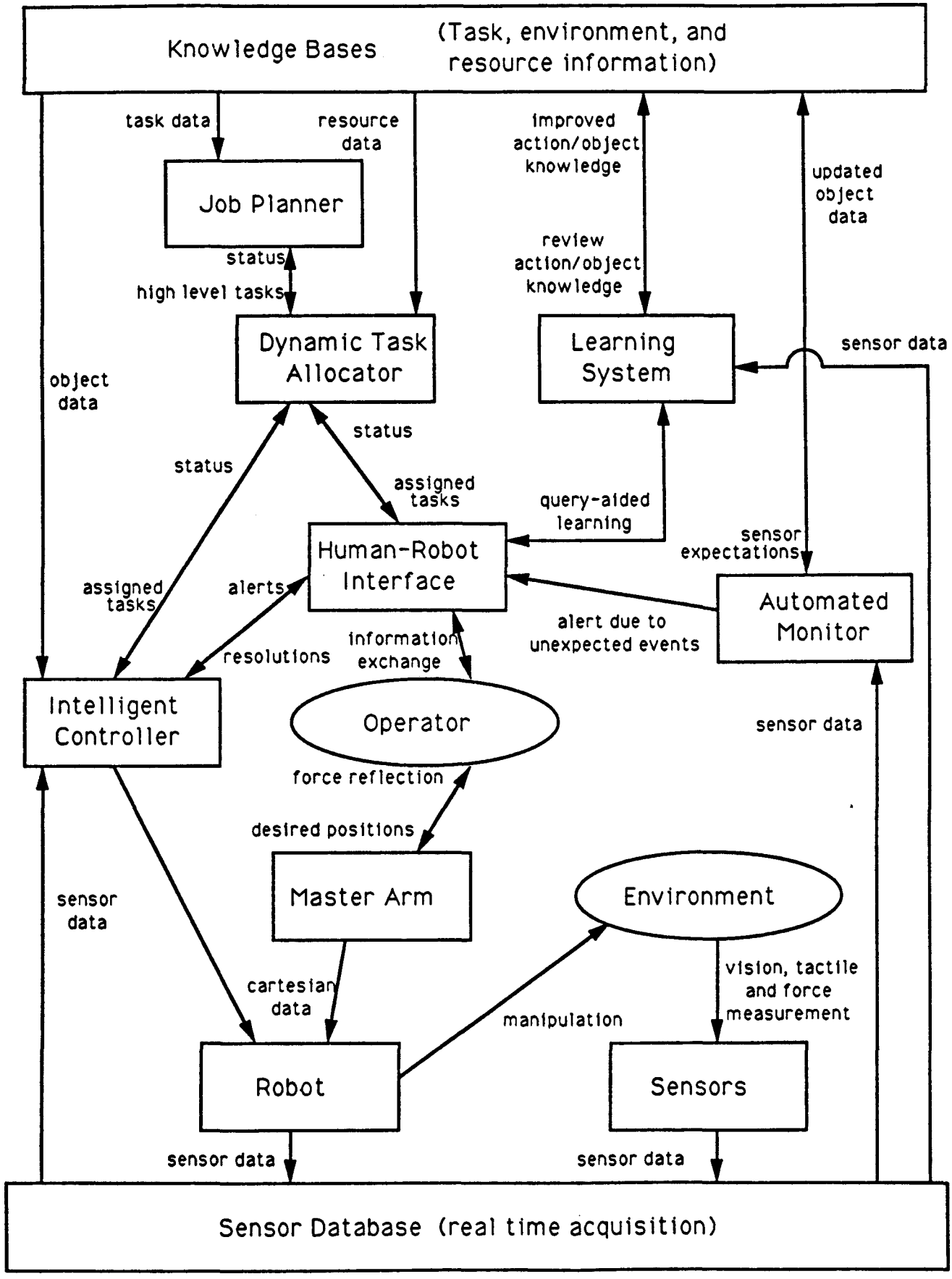


Figure 1. System Architecture for an ORNL Human-Robot Symbiotic System

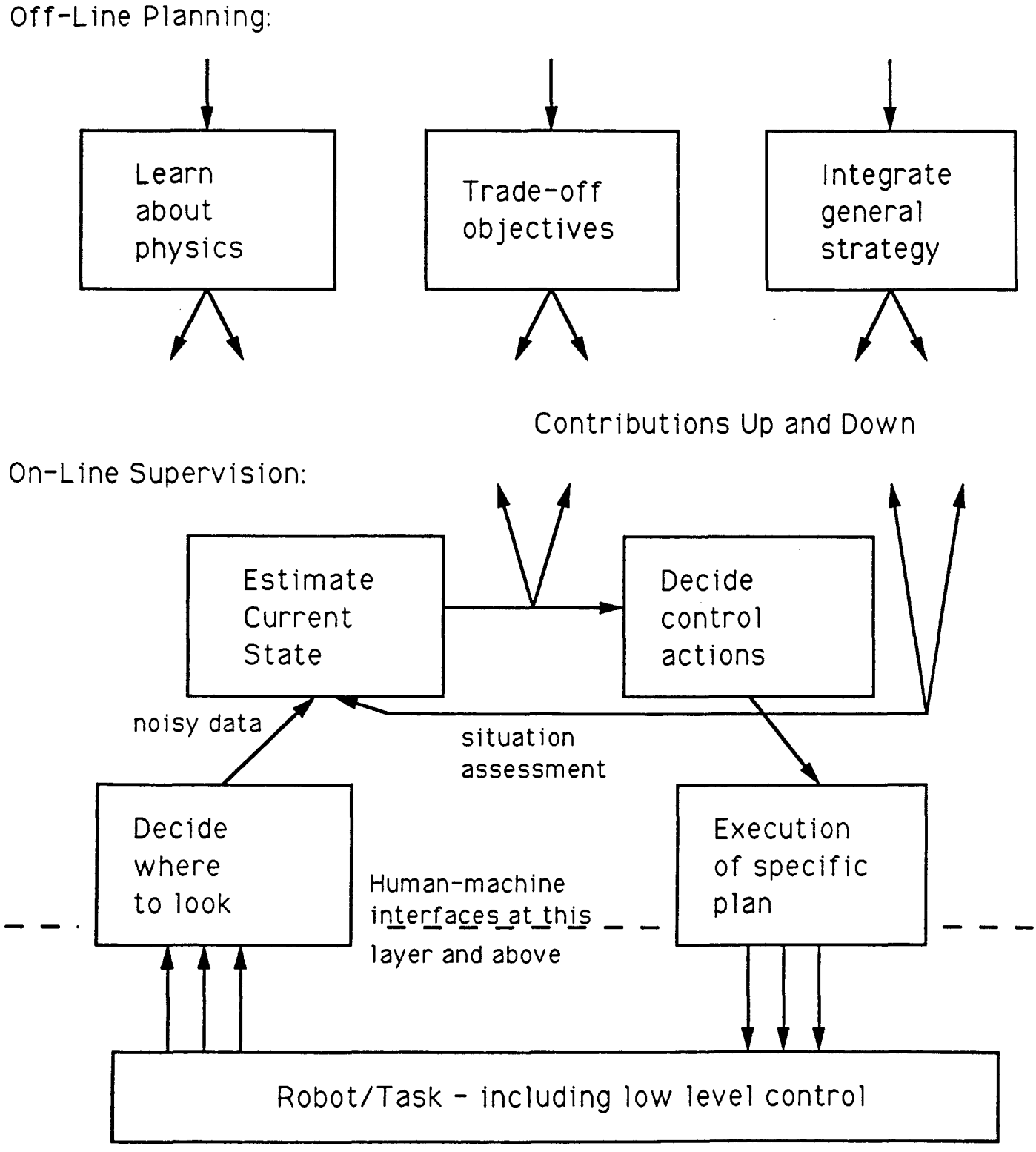


Figure 2. Architecture for a Human-Robotic System

Blank Page

Breakout Session: Machine Learning *via* Experience and Human Observation

December 6, 1988

Philip F. Spelt
Oak Ridge National Laboratory

Attendees:

Francois G. Pin — Oak Ridge National Laboratory — Chair
Philip F. Spelt — Oak Ridge National Laboratory — Reporter
Gerard de Saussure — Oak Ridge National Laboratory
Maria Gini — University of Minnesota
Claudio Gutierrez — University of Delaware
Scott Harmon — Robot Intelligence International
Susan Hruska — Jacksonville State University
Chuck Jorgensen — Johnson CSF
Avi Kak — Purdue University
Bill Knee — Oak Ridge National Laboratory
Thomas Mitchell — Carnegie-Mellon University
Eui H. Park — North Carolina A & T
Lynne Parker — Oak Ridge National Laboratory
Michael Rabins — Texas A & M
Ching-Long Shih — University of Kentucky
Harold P. Van Cott — National Acadamy of Sciences

Question: What are the most promising approaches toward providing the intelligence for a machine to learn new tasks through assimilation of experience and human observation?

Abstract

Considerable time and effort were spent to define the terms "Learning" and "Symbiosis" in the Human-Machine Symbiosis context. While the original question for the group was not directly answered, the issues raised by the 5 questions presented below should serve as areas of focus for research on human–machine symbiosis in the next few years. The utility of both inductive, data-driven learning systems and deductive, knowledge-driven systems was highlighted, and there was some suggestion that a blending of the two approaches would be good.

Initial "Charge to Group" by F. G. Pin — To answer the question posed to the group, it would be helpful to consider the following three questions which relate to or derive from that overall question:

1. What are the assets and liabilities of human–machine symbiotic systems?

2. What is learnable in the human–machine context?

3. What are the promising approaches, now and in the future?

As presented in the CESAR Human–Machine Symbiosis project (see Figure 1), machine learning in a Human–Machine Symbiotic system requires communication between an *Automated Observer* of human performance, and a *Human Operator*. Focus for this discussion session was on the relationship between those two components, as illustrated in the highlighted area of Figure 1. By implication, then, *Learning* in a Human–Machine Symbiotic system involves, in part, a transfer of knowledge from the human to the machine. This process must also include the *Automated Monitor* shown in Figure 1, to observe the overall performance of the entire system.

In attempting to answer the question presented to the group, a number of issues were raised. Considerable time was spent defining terms and exploring the limits of applicability of those terms. The material which follows summarizes and organizes the discussion around the major points made by the group.

I. Definition of Terms: Machine Learning and Symbiosis

A. Learning — The group spent considerable time and effort discussing what was meant by the term "Machine Learning," and even after a definition was adopted and recorded, the discussion periodically returned to this topic, indicating the volatile and ubiquitous nature of the term. A problem arose when the group, at times, forgot that the discussion actually should have been limited to "Learning" in a *Symbiotic system*. Consequently, at times the process ranged rather far and wide in the general area of learning. Some discussion concerned the function or use of *feedback* in learning, a topic one participant talked of in terms of "Open-Loop" (without feedback) and "Closed-Loop" (with feedback) learning. It was pointed out that the cybernetic view of the human body holds that it is a system of closed loops which are constantly monitored. As is frequently the case in discussions of Machine Learning, examples of both machine and biological (human) learning were cited which purported to show that one or another type was or was not suitably included in learning.

After the far-ranging discussion described above, the following definition was accepted by the group:

Learning is: "The acquisition of knowledge resulting in an improvement of performance [by the system] at some task."

Discussion of various implementations of machine learning followed, especially those using Artificial Neural Networks — the Connectionist approach. In the context of Neural Networks, learning consists of a Correction Function which is applied to the differences between an input vector, representing the real world, and an output vector from the Network, constituting the system's representation of the world state. There was some discussion of what "knowledge" the Neural Network starts with, and the point was made that NO learning system starts from scratch — with NO information or knowledge. As one participant put it, "The machine ain't going to make this stuff up by itself — someone has to put something in." In the case of Artificial Neural Networks, the system begins with a minimum of some randomly distributed non-zero weights in the matrix or matrices which map the input vector(s) to the output of the network. In knowledge-based systems, someone must program in either the information itself or the ability for the system to acquire that knowledge from its experience (a form of learning). One specialized form of knowledge-based systems is the Expert System, which is currently the major alternative to Neural Networks in AI research in the United States.

The issue of evaluation also is raised by this definition of learning. Improvement of performance requires some operation for measuring a change in performance from one time (measurement) to another, and such evaluation "closes the loop" for learning. The techniques for measuring improvement are situation and/or task specific. This issue relates closely to another issue that arises specifically in the context of Human-Machine Symbiosis — that issue is, what is the function of learning in such a system, what component(s) learn, and how does the overall performance of the system change. These issues will be discussed more fully in a later section.

B. Symbiosis — In his opening remarks to the group, F. G. Pin presented an illustration of symbiosis from the animal world — a humorous caricature of a hippopotamus with a bird on its back removing parasites from the skin of the hippopotamus. The definition of "symbiosis" which the group inferred from this model does not seem suitable for a Human-Machine system, primarily because of the implied dependence for survival of the two symbionts on each other. Such mutual dependence is not suited to Human-Machine systems. In contrast, the definition in Webster's II New Riverside University Dictionary does seem properly suited to this type of system:

> The relationship of two or more different organisms in a close association that may be but is not necessarily of benefit to each. (1984, p. 1172, emphasis added)

An alternative term for such a human-machine system might be Human-Machine Synergy —

> The action of two or more . . . organisms to achieve an effect of which each is individually incapable. (ibid, p. 1174)

For simplicity and for continuity with previous writing in this area, the term "symbiosis" will continue to be used here, but only in the sense conveyed by the dictionary definition presented above. Either of these terms, however, raises the issue of the role of the human in

the system, an issue which is discussed in a section to follow.

Another major issue raised by the concept of symbiosis is that of autonomy of the components of the system. Is autonomy a prerequisite for Symbiosis? That is, do both the human and the machine need the capacity to function entirely on their own in order for symbiosis to occur? It was the consensus of the group that Human-Machine Symbiosis requires some degree of autonomy on the part of the machine in order for it to be able to cooperate with the human. The key is to have a distributed system in which the machine and the human both make some decisions. There is serious question about how successful a Human-Machine system would be if the Intelligent Controller machine (see Figure 1) made all the decisions concerning task allocation, in a machine-controlled-human sense.

II. What Learning Techniques Are Available?

There appear to be several ways of "slicing the pie" to answer the question which heads this section. One way suggested was to contrast Inductive with Deductive learning systems. Inductive learning systems are Data-driven systems which learn from training examples (data). This type of learning is found in Artificial Neural Networks, which embody the connectionist approach to learning described above. Contrasting with data-intensive systems are those which can be characterized as Knowledge intensive. Knowledge-driven or concept-driven, these deductive systems learn rules for relating outside or new information to information already in the system. This type of system can be said to do feature extraction, in which the "important" features of the outside world are selected for processing in the system.

A question arose concerning whether Neural Networks are an implementation of data-driven learning and Expert Systems are, correspondingly, an implementation of the knowledge-driven approach. The group generally agreed that the situation is not so simple — that both implementations can and do use both approaches, but the two types naturally incline toward such a dichotomy. Possible uses of both these types of learning system in the Human-Machine Symbiosis context were considered. Inductive, data-driven systems appear to be useful for modeling the human operator, especially those characteristics which are difficult to quantify. Deductive, knowledge-driven systems seem suited for decision making, and for use in situations where heuristics are needed for coping with the situation. Deductive systems need to have enough of the proper kind of knowledge written in for them to function properly. Currently in the AI literature, there is recognition that connectionist systems are suited for low-level processing/learning, while knowledge-based systems are good for higher-level functions.

III. Role of Human in HMS Learning

In light of the preceding considerations, there was general agreement that there are 2 roles for humans in a Human-Machine Symbiotic System. One is to serve as a Model for the machine component to learn from. The human's activity in this case is to simply perform the task(s) while the machine part of the system observes and learns. If such a process were to be

automatic, then it would require the Automated Observer in the lower right of Figure 1 to monitor task execution and feed information back to the Learning System. The second role for humans in this type of system is that of Teacher/Consultant (Approver). In this situation, the human would monitor performance of the machine (robot) component and make corrections as needed. Here, the activity of the human would be active teaching or coaching of the machine component.

The preceding analysis makes a number of assumptions about the functioning and characteristics of the various components in the Human-Machine System. First, it assumes that the human is "superior" early in the performance of the tasks. This is often a valid assumption, however, as in many situations humans are capable of doing the entire set of tasks. A second assumption is that, as the system gains experience, the machine becomes a more important component — machine takes on more of the functions originally performed by the human. To the extent that this assumption is true, it expresses an important goal of the learning component of the symbiotic system.

The assumptions discussed above raise major issues. One concerns the question of whether, if the machine learns and takes over more of the tasks, does the overall system Improve? Usually, a machine is more brittle than a human — that is, the machine's performance is more subject to failure caused by relatively small fluctuations in operating conditions, to which a human could easily adapt but to which the machine cannot. Under conditions in which the result is to lower overall system performance, can we say that learning has occurred? Certainly, this would not be a desirable outcome.

Another question raised by one group member was "Do we really want a human in the loop?" The implication of this question is that if we can automate a system so that a machine can perform the tasks adequately, there is no need to include a human in the system. Related to this issue is the one of whether we should make machines to do things the way people do things, or whether there are other more efficient ways of having machines do things. While neither of these questions was answered, they are nevertheless important questions that need to be asked about each implementation of Human-Machine Symbiosis.

Finally, the group considered the question: Is the Complementarity of humans and machines something to build on? The overwhelming answer to this question was YES!!! However, the answer to the related question "Do we know how to do it?" is not so clear-cut. The point is that machines are not the same as people, and, as implied by the questions in the preceding paragraph, we need to explore ways to maximize the contributions of both machines and humans in these Human-Machine Systems. These considerations raise even more questions about the nature of symbiotic systems involving machines and humans. The following issues can serve as a source of research topics for some time to come:

1. What role should learning play in a Machine and Human symbiotic relationship?

2. Can learning improve system performance (as opposed to improved machine performance with overall lowered system performance)?

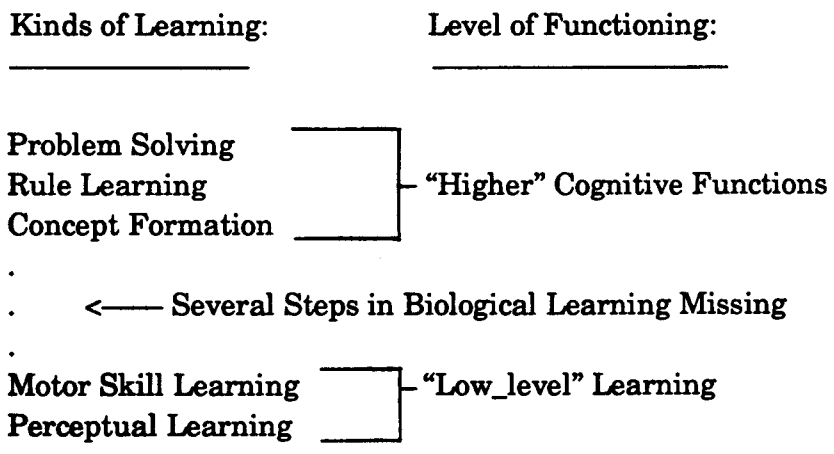
3. What is learnable? What kinds of subtasks could the machine learn, and what must (should) remain with the human?

4. Should the Human always be in control, or are there times when it is permissible for the machine to be in control?

5. When is Machine Learning useful?

The group was able to answer only some of these questions, and then only in a rather general way. In answer to question 5, machine learning is useful when not everything can be pre-programmed, partly because it can not be known a priori. In other words, a learning program is needed when one cannot specify all parameters and values in advance. The question of "What is learnable?" (number 3) was answered only by saying "To be learnable it must be observable." Clearly, all these issues need to be considered any time a symbiotic human-machine system is being designed. In considering question 4, it was noted that, if a system learns (in the sense of shifting tasks to the machine), then the system can, in turn, become the teacher of an untrained human. In other words, the question of who is in control may be answered by determining which component is the "expert" or master craftsman and which is the apprentice (see bottom of Figure 1).

One member pointed out that machine learning is still very much "in its infancy", compared to other areas of computer science, which is itself a relatively young science. Moreover, there has been a tendency for people in the machine learning field to develop unique terminology for concepts that have been used for some time in other fields, for example in the psychology of learning and memory. To help clarify thinking on the various topics of machine learning, the following Hierarchy of Learning was presented:



Machine Learning has tended to focus on the top three as being representative of "Intelligent" systems. The bottom two are becoming important for robotics, as attempts are made to create machines which can navigate in the environment and do things. Very little work in machine/ robot learning has explored the intermediate kinds of learning which are, nevertheless, very

important for intelligent behavior in biological systems, including humans.

IV. Discussion of Gains in next 2 Years for Machine Learning in HMS systems (needed &/or desired)

At the end of the session, the group was asked to briefly consider what gains might be expected in the next two years, or what problems need to be addressed. It was felt that there would be more work done on learning by machines from examples provided by people — learning from trained (observed) sequences, with generalization being an important element of the learning. An important component of this will be the ability of machines to observe whether the goal is reached, and make corrections in the operation when it is not.

Pessimism was expressed by one or two about machine learning as a long-lasting endeavor, pessimism which stems from comparing what even a human infant can learn with what we can make a machine learn. We don't know enough about how to create learning machines. One person felt that there is no such thing as learning in the strong sense of the word, that we can have only adaptive systems. Additional areas which the group felt are either important or interesting include adapting to changing environments (a system which learns about patterns of events in time), learning by extrapolating knowledge from one domain to another (generalization is important), and integration of sensory information, enabling a machine to discover new capabilities. We also need ways of interpreting sensor data.

V. Summary

As might be expected with a group of this size and diversity, full agreement on issues and terms was difficult to achieve. The group spent considerable time and effort to arrive at a definition of the term "Learning" in the Human-Machine Symbiosis context. Even after accepting one, discussion returned several times to that topic, although the original definition was left intact. The term "Symbiosis" was also defined, as a more restricted version of the group's initial view, which included some element of mutual dependence by the symbionts. While the original question for the group was not directly answered, the issues raised by the 5 questions presented in section III above should serve as areas of focus for those interested in research on Human-Machine Symbiosis in the next few years. The utility of both inductive, data-driven systems and deductive, knowledge-driven systems was highlighted, and there was some suggestion that a blending of the two approaches would be good.

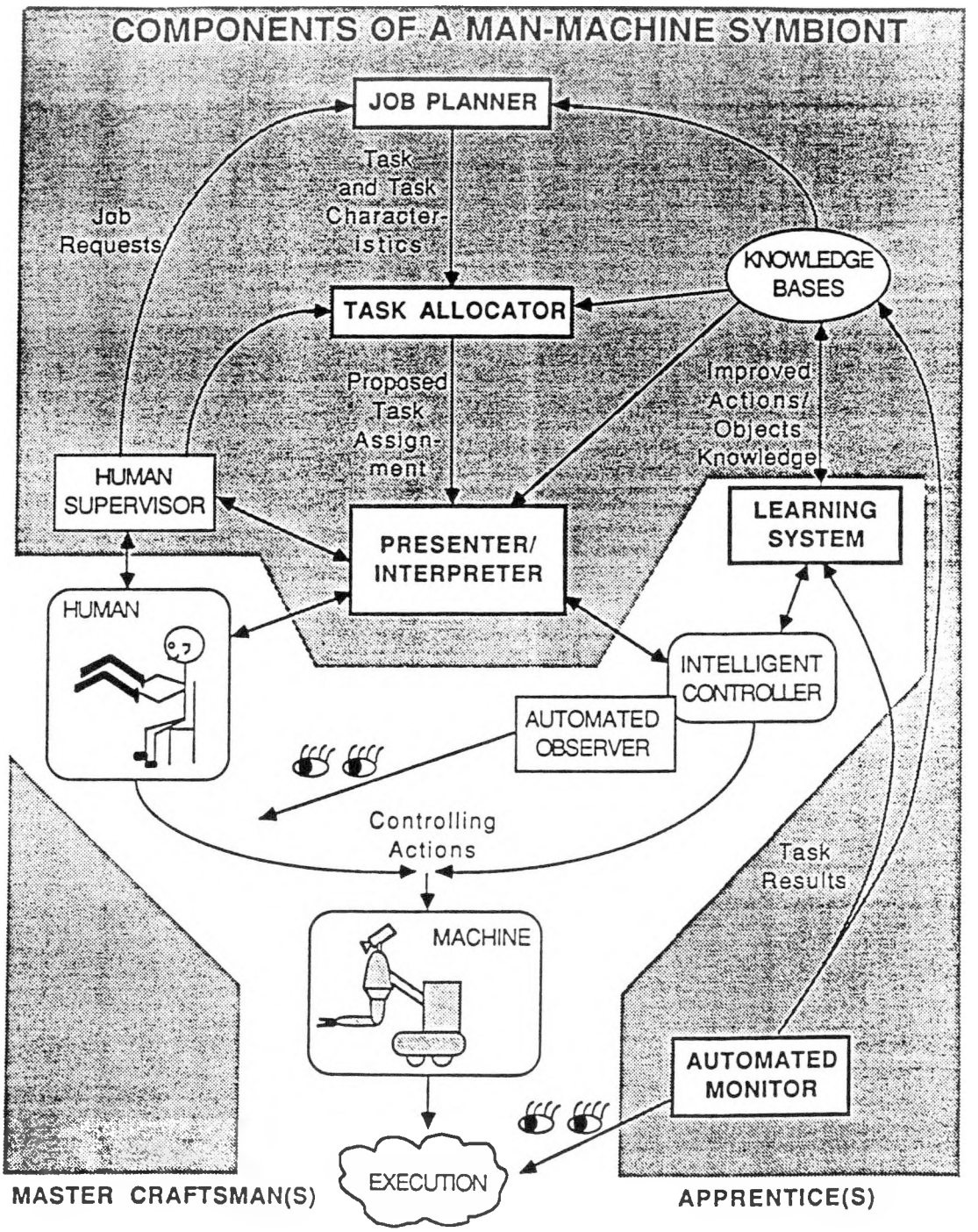


Figure 1. Man-Machine Symbiosis

Section 6: List of Participants

Blank Page

List of Participants

Dr. Marty Beckerman
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Kai-Hsiung Chang
Department of Computer Science and Engineering
Auburn University
Auburn, Alabama 36849

Dr. Prem Chopra
School of Engineering
University of Tennessee-Chattanooga
Chattanooga, Tennessee 37403

Dr. Kevin Corker
BBN Systems & Technologies Corporation
70 Fawcett Street
Cambridge, Massachusetts 02178

Mr. Fred Depiero
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Gerard DeSaussure
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Maria L. Gini
Department of Computer Science
4-192 EE/CSA Building
200 Union Street, SE
Minneapolis, Minnesota 55455

Professor James H. Graham
Department of Engineering Mathematics and Computer Science
University of Louisville
Louisville, Kentucky 40292

Dr. James L. Gumnick
University Relations Office
Oak Ridge Associated Universities
Oak Ridge, Tennessee 37830

Dr. Claudio Gutierrez
Professor of Computer and Information Sciences
University of Delaware
Newark, Delaware 19716

Dr. Bill Hamel
Instrumentation and Controls Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Jia-Yuan Han
Department of Electrical Engineering
Southern Illinois University
Carbondale, Illinois 62901-6603

Dr. Scott V. Harmon
Robot Intelligence International
4660 Long Branch Avenue
San Diego, California 92107

Ms. Susan Hruska
Jacksonville State University
Jacksonville, Alabama 36265

Dr. Thomas Hutchinson
Computer Science Department
University of Virginia
Charlottesville, Virginia 22904

Dr. Chuck Jorgensen
Thomson-CSF, Inc.
630 Hansen Way, Suite 250
Palo Alto, California 94307

Dr. Avi Kak
Department of Electrical Engineering
Purdue University
Lafayette, Indiana 47907

Dr. Kazuhiko Kawamura
Department of Electrical Engineering
Box 1674, Station B
Vanderbilt University
Nashville, Tennessee 37235

Mr. Bill Knee
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Charles Kring
Fuel Recycle Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Wayne Manges
Instrumentation and Controls Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Reinhold Mann
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Christine Mitchell
Man-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

Dr. Thomas Mitchell
Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Dr. Amit Mukerjee
Computer Science Department
Texas A&M University
College Station, Texas 77843

Dr. Eui H. Park
Department of Industrial Engineering
North Carolina A&T State University
Greensboro, North Carolina 27411

Ms. Lynne Parker
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37830

Dr. Francois Pin
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37830

Dr. Michael J. Rabins
Department of Mechanical Engineering
The Texas A&M University
100 Engineering Physics Building
College Station, Texas 77843-3123

Dr. Thomas Sheridan
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Dr. Ching-Long Shih
Center for Robotics and
Manufacturing Systems
University of Kentucky
Breckinridge Hall
Lexington, Kentucky
40506-0056

Dr. Philip Spelt
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. James S. Tulenko
202 Nuclear Science Center
The University of Florida
Gainesville, Florida 32601

Dr. Harold P. Van Cott
National Academy of Sciences
2101 Constitution Ave., NW
Washington, DC 20418

Dr. Chuck Weisbin
Engineering Physics and
Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831

Dr. Yuan F. Zheng
Department of ECE
Clemson University
Clemson, South Carolina
29634

Blank Page

Section 7: Biosketches

Blank Page

Biosketches

MARTIN BECKERMAN

Martin Beckerman, a staff scientist in Oak Ridge National Laboratory's Engineering Physics and Mathematics (EPM) Division, conducts research in sensor modelling, sensor fusion, and image model building. Prior to joining the EPM Division he served as a research associate professor of physics at the University of Tennessee and as a research scientist and principal investigator in the Laboratory for Nuclear Science of the Massachusetts Institute of Technology. He was an Institute Fellow in the Department of Nuclear Physics of the Weizmann Institute of Science and has been a consultant in heavy-ion experimental physics to the Physics Division and the Kellogg Radiation Laboratory at Caltech. He has authored over 70 publications in robotics and experimental and theoretical nuclear physics.

KAI-HSIUNG CHANG

Kai-Hsiung Chang received his diploma in electrical engineering from the Taipei Institute of Technology in 1977 and his M.S. and Ph.D. in electrical and computer engineering from the University of Cincinnati in 1982 and 1986, respectively. Currently, he is an assistant professor of computer science and engineering at Auburn University. His present interests include knowledge-based systems, manufacturing systems, and computer vision. He is a member of the IEEE and AAAI. His address is the Department of Computer Science and Engineering, Auburn University, Auburn, AL 36849.

GERARD DE SAUSSURE

Gerard de Saussure, a staff scientist in Oak Ridge National Laboratory's Engineering Physics and Mathematics Division, has worked for over 30 years in experimental nuclear physics. Currently, he splits his time between nuclear physics and research in strategy planning and machine intelligence at the Center for Engineering Systems Advanced Research and is also an honorary professor at the University of Tennessee. He received his Ph.D. in experimental physics from Massachusetts Institute of Technology.

MARIA L. GINI

Maria L. Gini is an associate professor at the University of Minnesota, Department of Computer Science, in Minneapolis.

She has been a research associate at the Department of Electronics, School of Engineering, Polytechnic of Milan, Italy, and a visiting research associate at the Artificial Intelligence Laboratory at Stanford University.

Her research interests are in the area of artificial intelligence and robotics. She is mainly interested in programming robots and making robots understand more about the environment in which they operate. She has worked on automatic error detection and recovery for assembly robots, navigation of robots in unknown environments with moving obstacles, integration of planning with execution, and programming of robots. She is author of several publications on those subjects.

JAMES H. GRAHAM

James H. Graham [redacted] He earned his bachelor's degree in electrical engineering from the Rose Polytechnic Institute and his master's and doctoral degrees in electrical engineering from Purdue University. He is a registered professional engineer and has worked as a product design engineer with General Motors Corporation. He has served on the faculty at Rensselaer Polytechnic Institute and is presently an associate professor of computer science at the University of Louisville. His research interests include robotics, artificial intelligence, and parallel computation.

CLAUDIO GUTIERREZ

Claudio Gutierrez is a professor of artificial intelligence at the Department of Computer and Information Sciences, University of Delaware.

He is a citizen of Costa Rica, where he was president of the leading institution of higher learning, the University of Costa Rica, and taught Philosophy and Computer Science for more than two decades.

He has a Ph.D. in philosophy of science from the University of Chicago and is "licenciado" in Law and in History of the University of Costa Rica. He received the Guggenheim Fellowship (1966), was the Langston Hughes Professor at the University of Kansas (1982), and was a distinguished visiting professor at the University of Delaware (1981–1983).

He is the author of several books on philosophy of science, epistemology, ethics, and social impact of computing. During the last eight years, he has been developing a symbiotic system

for administration (AISA). In the design of such system, he has taken advantage of his experience as a former university administrator and the analytical insights of his philosophical education.

WILLIAM R. HAMEL

Dr. William R. Hamel is head of the Telerobotic Systems Section in the Instrumentation and Controls Division at the Oak Ridge National Laboratory. As head of the Telerobotic Systems Section, he directs a research team with expertise in robot/teleoperator systems engineering, manipulator controls, real-time digital controls, servo/digital electronics and robotic sensing. He has a B.S.M.E. from West Virginia University, a M.S.M.E. from Oklahoma State University, and a Ph.D. from the University of Tennessee. He is a member of the Sigma Xi, Tau Beta Pi and Phi Kappa Phi honoraries. Dr. Hamel's research interests include high performance robot design, manipulator dynamics and control, robot sensor integration, and human-machine interactions. He has written numerous reports and papers in the areas of nuclear remote technology and the advancing field of robotics. He maintains memberships in the ASME, IEEE, and Robotics International.

JIA-YUAN HAN

Jia-Yuan Han received an M.S. in applied mathematics from Ohio State University in 1983, and a Ph.D. in electrical engineering from the Ohio State University in 1986. His research interests are real-time computation systems and their application in robotics, signal processing and pattern recognition, task planning, decomposition, scheduling, fault-tolerance computing, robotics, and neural networks.

SCOTT Y. HARMON

Scott Y. Harmon is owner of Robot Intelligence International, a robotics research and development firm based in San Diego, California, USA. With over 16 years research experience, he has concentrated for the past 13 years on advanced robot system and autonomous system research. An internationally recognized authority on mobile robots, autonomous systems, and cooperating robots, Mr. Harmon has authored over 40 technical publications on robot system integration, mobile robots, sensor data fusion, distributed robotics, military robots, robot computing architectures, and robot planning. He has conducted two extensive surveys of European robotics and teleoperated system research, coorganized the 1984 ONR European Workshop on Robotics, and organized the 1987 NATO Advanced Research Workshop on Mobile Robot Implementation. He has been an invited contributor to the NATO Advanced Research Workshops on Machine Intelligence for Robotic Applications, on Mobile Robots and on Highly Redundant Sensing Systems, as well as the 1986 DARPA Workshop on Blackboard Systems for Robot Perception and Control. He has also been invited to participate in the 1989 NATO Closing Workshop for Advanced Research Workshop Directors on Sensory Systems for Robotic Control and on a panel on Robot Navigation for the 1989

International Joint Conference on Artificial Intelligence. Mr. Harmon originated, managed and contributed technically to the Ground Surveillance Robot project, an effort to develop an autonomous ground vehicle which could cross unknown natural terrain. He is currently developing the autonomous ground navigation system for the Mars Rover Sample Return project.

SUSAN I. HRUSKA

Susan Hruska is assistant professor at the Department Computer Science and Mathematics, Jacksonville State University, Jacksonville, Alabama. She received the B.S. and M.S. degrees from Auburn University and the Ph.D. from University of Alabama in Birmingham.

KAZUHIKO KAWAMURA

Kazuhiko Kawamura (Ph.D., Michigan) is a professor and the director of graduate studies of electrical engineering at Vanderbilt University. He is also Associate Director of the Center for Intelligent Systems, a multi-disciplinary research center engaged in studies in applied artificial intelligence at the School of Engineering. He directs research projects in intelligent robotics, parallel computer vision systems, intelligent tutoring systems, and knowledge-based risk assessment systems.

His past positions include an instructor at the University of Michigan, Dearborn, research specialist at Ford Motor Company, principal systems planner at Battelle Columbus Laboratories, and invited professor at Kyoto University, Kyoto, Japan.

AVI KAK

Avi Kak is a professor of electrical engineering at Purdue University. He is in charge of the Robot Vision Laboratory with his current research in high level planning and sensing in robotics.

H. E. KNEE

Mr. Knee is the group leader of the Cognitive Science and Human Factors Group at the Oak Ridge National Laboratory (ORNL). This group is composed of an interdisciplinary team of engineers, psychologists, and computer scientists that carry out applied research and development, applications and basic research in human factors, systems reliability, systems engineering/analysis, artificial intelligence, and cognitive science. Mr. Knee holds B.S. (1974) and M.S. (1976) degrees in Nuclear Engineering from the University of California at Los Angeles, and an M.B.A. (1986) with an emphasis in management from the University of Tennessee. Mr. Knee has been with ORNL since 1976 and has been involved with human factors research for over a decade. His research efforts over this time period involved: (1) the

development (for the Nuclear Regulatory Commission) of the MAPPS (Maintenance Personnel Performance Simulation) computer simulation model of nuclear power plant maintainer activities, (2) the development of the Centralized Reliability Data Organization (CREDO), an international advanced reactor reliability, availability, and maintainability data system and data analysis center, and (3) initial development of a cognitive model of nuclear power plant operator behavior entitled INTEROPS (Integrated Reactor Operator/System). Mr. Knee's current research interests involve human behavioral/cognitive modeling, human/system interaction in advanced system design, human/system reliability, and cognitive science

WAYNE W. MANGES

Wayne has been involved in real-time computer activities in the Instrumentation and Controls Division of Oak Ridge National Laboratory (ORNL) for eleven years. He currently leads the Telerobotic Sensors and Electronics Group. Before his current position at ORNL, he was a high school physics and chemistry teacher. He holds a B.S. and an M.S. in pure science as well as a B.S. and M.S./EE. His interests include real-time computer hardware and software architecture, control system architectures, and software quality measurement and control. He has several publications in the areas of hierarchical process control, management of large software projects, and real-time networking of heterogenous computer architectures.

REINHOLD C. MANN

Reinhold C. Mann received a Diplom-Mathematiker degree (M.S. in mathematics) in 1977, and a Dr. rer. nat. degree (Ph.D.) in physics in 1980 from the Johannes Gutenberg University in Mainz, Federal Republic of Germany (F.R.G.) From 1978 until 1980 he was a research associate in the Biophysics Department at Mainz University and a consultant with the Laser and Optics Group at Battelle Institute in Frankfurt, F.R.G., in the areas of digital image analysis and pattern recognition. In 1980 he joined the Image Analysis Group at the Fraunhofer Institute for Data and Information Processing in Karlsruhe, F.R.G. He was awarded a Feodor-Lynen Fellowship by the Alexander von Humboldt Foundation in Bonn, F.R.G., which allowed him to spend 1981 and 1982 as a visiting scientist at the Oak Ridge National Laboratory (ORNL), working on biomedical applications of pattern recognition and image analysis. He has been a staff member at ORNL since 1983 and joined the Robotics and Intelligent Systems Program in 1986. He has been leader of the Advanced Computing and Integrated Sensor Systems Group at ORNL since 1987. He is an Adjunct Professor in the Computer Science Department at the University of Tennessee in Knoxville. His research interests include computer vision, multisensor integration, pattern recognition, and concurrent computing. He has authored over 20 reports and publications.

CHRIS MITCHELL

Chris Mitchell received her Ph.D. in industrial and systems engineering from Ohio State University. She was a member of George Mason University's Decision Sciences Faculty. In 1984, she accepted a position with Georgia Tech School of Industrial and Systems Engineer

ing and Center for Human–Machine Systems Research. Her research interests are in the areas of models of human–computer interaction in supervisory control systems and the design of teams of decision makers that combine human and computer components.

TOM M. MITCHELL

Tom M. Mitchell is a professor of computer science at Carnegie Mellon University and an affiliated faculty of the Robotics Institute. He earned his B.S. degree (1973) from Massachusetts Institute Technology and his M.S. (1975) and Ph.D. (1978) degrees from Stanford University. He taught in the Computer Science Department at Rutgers University from 1978 until moving to Carnegie Mellon in 1986. In 1983 he received the IJCAI Computers and Thought award in recognition of his research in machine learning and in 1984 an NSF Presidential Young Investigator Award. His current research focuses on developing robots that learn and on general architectures for problem solving and learning. His current address is Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

EUI PARK

Dr. Eui Park is an associate professor of industrial engineering at North Carolina A&T State University and currently is conducting the quality control and manufacturing programs in the department. Dr. Park earned his Ph.D. from Mississippi State University in 1983. His research interest areas are production systems design and artificial intelligence (AI) applications. He currently has two research contracts with NASA and U.S. Air Force in the area of telerobotics. In addition to his teaching and research experience, Dr. Park spent several years as a senior engineer in Engineering Computing Systems at Boeing Commercial Airplane Company.

LYNNE E. PARKER

Lynne E. Parker, a staff scientist in Oak Ridge National Laboratory's Robotics and Intelligent Systems Program, conducts research in human–machine symbiosis and related artificial intelligence areas, including job planning, dynamic task allocation, and automated monitoring. Prior to joining ORNL, she worked as a Computer Systems Analyst with Martin Marietta Energy Systems. She received the B.S. degree in computer science from Tennessee Technological University and the M.S. degree in computer science from the University of Tennessee, Knoxville. Besides human–machine symbiosis, her current research interests are multiple robot cooperation and autonomous mobile robot navigation.

FRANCOIS G. PIN

Francois G. Pin heads the Machine Reasoning and Automated Methods Group at Oak Ridge National Laboratory. He also leads the Machine Intelligence and Advanced Computing Sys-

tems activities of the Robotics and Intelligent Systems Program and is a principal investigator of the Center for Engineering Systems Advanced Research. His technical interests include high-level planning, reasoning, problem solving and learning for autonomous mobile systems, and man–machine symbionts. He received his M.S. and Ph.D. in mechanical engineering from the University of Rochester, New York.

MICHAEL J. RABINS

Michael J. Rabins joined the Texas A&M University Mechanical Engineering Department as Head in February 1987. Prior to that he was at the Wayne State University College of Engineering in September, 1977, first as Chairman of the Mechanical Engineering Department, and then from June, 1985 he served as associate dean for engineering research and graduate programs at Wayne. Prior to 1977, he was on the Polytechnic Institute of New York faculty from June 1970 as a professor of system engineering and director of the System Engineering Program in the Department of Operations and System Analysis. Between 1960 and 1970 he served as assistant and associate professor of mechanical engineering at New York University. He plays an active role in the American Society of Mechanical Engineers (ASME), of which he is a Fellow, where he was (founding) Editor of the ASME Quarterly, *Dynamic Systems, Measurement and Control* from 1970 to 1973; and chairman of the Executive Committee of the Automatic Control Division of the A.S.M.E. in 1975. In 1977 he became an elected member-at-large of the ASME Policy Board, Communications, and served as ASME vice-president of communications from 1981 through 1984. Currently, he serves as chairman of the Technology Opportunity and Planning Committee of the American Society of Mechanical Engineers Board on Research and Technology Development. He is also a member of ASEE, ISA, and an elected member of the College of Fellows of the Engineering Society of Detroit. He is the immediate past-president of the American Automatic Control Council.

THOMAS B. SHERIDAN

Thomas B. Sheridan [redacted] He received the B.S. degree from Purdue University, West Lafayette, IN (1951), the MS degree from University of California at Los Angeles (1954), and the ScD degree from Massachusetts Institute of Technology (MIT), Cambridge, MA (1959). From 1951–1953 he was a research and development officer in the Air Force and a rated parachutist.

For most of his professional career he has remained at MIT, where he was assistant professor, associate professor, and professor of mechanical engineering and is now professor of engineering and applied psychology and director of the Man-Machine Systems Laboratory. He has also served as a visiting faculty member at University of California, Berkeley, Stanford University and Technical University of Delft, Netherlands. Dr. Sheridan's research and teaching activities include design, control, modeling and human factors experimentation of telerobotic systems for space and undersea, operation of nuclear power plants and commercial aircraft, individual and group decision processes, and social effects of automation. Dr. Sheridan has also been associated with the MIT programs in technology and policy and in science, technology, and society.

Dr. Sheridan served as president of the IEEE Systems, Man and Cybernetics Society, Editor of *IEEE Transactions on Man-Machine Systems*, and is an IEEE Fellow and Centennial Medalist. He is also a Fellow of the Human Factors Society and recipient of their Paul M. Fitts Award for contributions to education. He is coauthor of *Man-Machine Systems* (MIT Press, 1981,84; USSR, 1981) and coeditor of *Monitoring Behavior and Supervisory Control* (Plenum, 1976). He has served on the editorial boards of Automatica, Advanced Robotics, Robotics and Computer Integrated Manufacturing, Computer-Aided Design, and Technological Forecasting and Social Change. He is coholder of two patents.

Dr. Sheridan has served on various advisory committees of the National Institutes of Health, Office of Technology Assessment, NRC, NSF, and NASA. He was chairman of the National Research Council's Committee on Human Factors and currently is a member of the NRC Committee on a Commercially Developed Space Facility. He is also a member of the Nuclear Regulatory Commission's Research Review Committee and NASA's Oversight Committee for the Flight Telerobotic Servicer. He is listed in *Who's Who in America* and similar listings.

CHING-LONG SHIH

Ching-Long Shih [redacted] He received B.S. and M.S. degrees in control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1980 and 1984. He received his Ph.D. in Electrical Engineering from Ohio State University of 1988. He is currently with the Center for Robotics and Manufacturing Systems, University of Kentucky. His main interests are in the areas of robotics, walking machines, and artificial intelligence

PHILIP F. SPELT

Philip F. Spelt, Ph.D., a consulting cognitive scientist in Oak Ridge National Laboratory's Engineering Physics and Mathematics Division, has worked during the past year to develop the learning and inferencing system components for the autonomous mobile robot at the Center for Engineering Systems Advanced Research Laboratory. He is on extended leave from Wabash College, Crawfordsville, Indiana, where he is Professor of Psychology. He received his B.A. in psychology from Grinnell College, Iowa, and his M.Sc. and Ph.D. degrees from the University of Pittsburgh in experimental psychology. He has done work on animal learning and memory and on computer simulation during his 20-year career.

JAMES S. TULENKO

Professor James S. Tulenko is chairman of the Department of Nuclear Engineering Sciences at the University of Florida. He holds degrees from Harvard and Massachusetts Institute Technology in applied physics and nuclear engineering. He is a principal investigator for the University of Florida in the Department of Energy University Program in Robotics for Advanced Reactors. He is also chairman of the Research Committee of the Utility/Manufac

turers Robotic Users Group. He is the author of over forty papers on nuclear engineering, workstations, and robotic modeling

HAROLD P. VAN COTT

Harold P. Van Cott is the study director for the Committee on Human Factors of the National Academy of Science/National Research Council. Previous positions include vice president and director, Systems Operability Division, Essex Corporation; chief scientist, BioTechnology, Inc.; division chief, National Engineering Laboratory, National Bureau of Standards; and director, Institute for Research on Human Performance, American Institutes for Research. Common to all of these positions is research related to the proper use of technology in relation to human needs, capabilities and limitations. Dr. Van Cott is a Fellow of the APA, AAAS, Washington Academy of Science, and the Human Factors Society. He received a Ph.D. in experimental psychology from the University of North Carolina.

CHARLES R. WEISBIN

Charles R. Weisbin is the director of the Robotics and Intelligent Systems Program at Oak Ridge National Laboratory (ORNL), where he is responsible for directing robotics and related artificial intelligence and parallel computing projects for the Department of Energy (DOE) and the Department of Defense, and other sponsors. He is also the Director of the Center for Engineering Systems Advanced Research and head of the Mathematical Modeling and Intelligent Control Section at ORNL. Dr. Weisbin serves as an associate editor of *IEEE Expert*, as a member of Martin Marietta Corporation Artificial Intelligence Steering Committee, and as DOE representative to the Department of Defense, Joint Directors of Laboratories, Robotics Technology Panel. He received his Eng.Sc.D. from Columbia University in nuclear engineering. His current research interests include robotics, concurrent computation, machine intelligence, decision making, sensitivity and uncertainty analysis, and human–machine symbiosis.

YUAN F. ZHENG

Yuan F. Zheng received his B.S. degree from Tsinghua University, Beijing, China, and M.S. and Ph.D. degrees from Ohio State University. After completing his Ph.D. degree in 1984, he joined Clemson University at Clemson, South Carolina. From 1984 to 1987, he was an assistant professor and became an associate professor in 1987. Dr. Zheng's research interests include a number of aspects in robotics and automation, such as robot arms for manipulation and robot legs for walking. Since 1984, he has made significant contributions in the coordination of multiple robot arms and biped walking robots. In recognition of his contributions, he was selected to receive the Presidential Young Investigator Award in 1987.

Blank Page

Section 8: ORNL Distribution

Blank Page

ORNL Internal Distribution

1. B. R. Appleton
2. S. M. Babcock
3. D. L. Barnett
4. M. B. Beckerman
5. P. F. R. Belmans
6. L. A. Berry
7. T. W. Burgess
9. B. L. Burks
10. R. J. Carter
11. J. C. Culioli
12. F. C. Davis
13. F. W. Depiero
14. G. de Saussure
15. B. G. Eads
16. J. R. Einstein
17. C. W. Glover
18. W. R. Hamel
19. J. H. Han
20. J. N. Herndon
21. J. P. Jones
22. S. M. Killough
23. H. E. Knee
25. D. P. Kuban
26-30. F. C. Maienschein
31. W. W. Manges
32. R. C. Mann
33. S. A. Meacham
34. J. R. Merriman
35. E. M. Oblow
36. P. J. Otaduy
37-41. L. E. Parker
42-46. F. G. Pin
47. D. B. Reister
48. J. T. Robinson
50. S. L. Schrock
51. J. C. Schryver
52. K. Setoguchi
53. T. E. Shannon
54. P. F. Spelt
55. F. J. Sweeney
56. M. A. Terranova
57. R. E. Urhig
58. K. Vancleave
59-63. C. R. Weisbin
64. J. D. White
5. B. A. Worley
66. H. R. Yook
67. A. Zucker
68. J. J. Dorning (Consultant)
70. Central Research Library
71. ORNL Technical Library Document Reference Section
72-73. Laboratory Records Dept.
74. Laboratory Records - RC
75. ORNL Patent Office
76. EPMD Reports Office

ORNL External Distribution

77. Office of Assistant Manager, Energy Research and Development, Department of Energy, Oak Ridge Operations, P.O. Box 2001, Oak Ridge, TN 37831

78. J. K. Aggarwal, University of Texas, Department of Electrical and Computer Engineering, Austin, TX 78712

79. Moonis Ali, University of Tennessee Space Institute, Tullahoma, TN 37388

80. Harry Alter, Division of Advanced Technology Development, 19901 Germantown Road, MS-542, Germantown, MD 20874

81. Michael A. Arbib, University of Southern California, Los Angeles, CA 90089-0782

82. Ruzena Bajcsy, Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104

83. Jacob Barhen, Jet Propulsion Laboratory, 4800 Oak Grove Dr., Pasadena, CA 91109

84. S. Baron, BBN Laboratories, Inc., 10 Moulton Street, Cambridge, MA 02238

85. Antal K. Bejczy, Robotics and Teleoperator Group, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr., Pasadena, CA 91109

86. George Bekey, Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782

87. Leo Beltracchi, Division of Reactor and Plant Systems, Office of Nuclear Regulatory Research, US NRC, MS NL/N-316, Washington, DC 20555

88. Thomas O. Binford, Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Stanford, CA 94305

89. Wayne Book, Department of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332

90. John Brewer, Mechanical Engineering Department, Louisiana State University, Baton Rouge, LA 70803-6413

91. Roger Brockett, Division of Engineering and Applied Physics, Harvard University, Pierce Hall, Cambridge, MA 02138

92. Rodney Brooks, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139

93. Reggie J. Caudill, Department of Mechanical Engineering, College of Engineering, Drawer ME, Tuscaloosa, AL 35487-2998

94-98. Kai-Hsiung Chang, Department of Computer Science and Engineering, 107 Dunstan Hall, Auburn University, Auburn, AL 36849-5347

99. Prem Chopra, School of Engineering, University of Tennessee-Chattanooga, Chattanooga, TN 37403

100. James S. Coleman ER-15, Director of Engineering and Geosciences, Office of Basic Energy Systems, U.S. Department of Energy, Washington, DC 20545

101. Lynn Conway, Associate Dean of College of Engineering, Chrysler Center, University of Michigan, Ann Arbor, MI 48109-2092

102. Kevin Corker, BBN Systems and Technologies Corporation, 70 Fawcett Street, Cambridge, MA 02178

103. Rui J. P. deFigueiredo, George R. Brown School of Engineering, Department of Electrical and Computer Engineering, P.O. Box 1892, Rice University, Houston, TX 77251

104. Max Donath, Department of Mechanical Engineering, University of Minnesota, 111 Church St., SE, Minneapolis, MN 55455

105. Joseph F. Engelberger, Transitions Research Corporation, 15 Durant Ave., Bethel, CT 06801

106. Bernard Espiau, IRISA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France

107. Gerard Fraize, UGRA, CEA CEN/Fontenay aux Roses, Fontenay aux Roses, France

108. Grafkan Galanos, Chairman, Department of Electrical Engineering, Southern Illinois University at Carbondale, Carbondale, IL 62901

109. Maria L. Gini, Department of Computer Science, 4-192 EE/CSA Building, 200 Union Street, SE, Minneapolis, MN 55455

110. Ralph Gonzalez, Department of Electrical and Computer Engineering, Ferris Hall, The University of Tennessee, Knoxville, TN 37996-2100

111 James H. Graham, Department of Engineering Mathematics and Computer Science, University of Louisville, Louisville, KY 40292

112. William A. Gruver, College of Engineering, Center for Robotics and Manufacturing Systems, University of Kentucky, Lexington, KY 40506-0056

113. James L. Gumnick, Office of University and Industrial Programs, Oak Ridge Associated Universities, Oak Ridge, TN 37830

114. Claudio Gutierrez, Professor of Computer and Information Sciences, University of Delaware, Newark, DE 19716

115. P. M. Haas, 10615 Alameda Drive, Knoxville, TN 37922

116. Capt. Samuel Hagins, Aircraft Launch and Recovery Branch, AFWAL/FDEMB, Wright-Patterson AFB, OH 45433-655

117. Ernest L. Hall, Department of Mechanical and Industrial Engineering, MS-72, University of Cincinnati, Cincinnati, OH 45221

118. Jia-Yuan Han, Department of Electrical Engineering, Southern Illinois University, Carbondale, IL 62901-6602

119. Scott V. Harmon, Robot Intelligence International, 4660 Long Branch Ave., San Diego, CA 92107

120. Ewald Heer, 5329 Crown Ave., La Canada, CA 91011

121. Marty Herman, Industrial Systems Division, National Bureau of Standards, Building 220/B124, Gaithersburg, MD 20899

122. David B. Hertz, Director, Intelligent Computer Systems, Research Institute, University of Miami, P.O. Box 248235, Coral Gables, FL 33156

123. Susan Hruska, Jacksonville State University, Jacksonville, AL 36265

124. Thomas Hutchinson, Computer Science Department, University of Virginia, Charlottesville, VA 22904

125. Sitharama S. Iyengar, Associate Professor, Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803-4020

126. Michael Jackson, Bldg. 4708, Room 206, Marshall Space Flight Center, AL 35806

127. Marilyn Jones, Department of Electrical Engineering, Virginia Tech, Blacksburg, VA 24061

128. Chuck Jorgensen, Thomson-CSF, Inc., 630 Hansen Way, Suite 250, Palo Alto, CA 94307

129. Avi Kak, Department of Electrical Engineering, Purdue University, Lafayette, IN 47907

130. Abraham Kandel, Florida State University, Inst. for Expert Systems and Robotics, Computer Science Department, LOV 206, Tallahassee, FL 32306-4019

131. Takeo Kanade, The Robotics Institute, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213

132. Kazuhiko Kawamura, Department of Electrical Engineering, Box 1674, Station B, Vanderbilt University, Nashville, TN 37235

133. Tarek M. Khalil, University of Miami, Department of Industrial Engineering, College of Engineering, P.O. Box 248294, University of Miami, Coral Gables, FL 33124

134. Jung H. Kim, Department of Electrical Engineering, North Carolina A&T State University, Greensboro, NC 27411

135. T. T. Lee, College of Engineering, Center for Robotics and Manufacturing Systems, University of Kentucky, Lexington, KY 40506-0056

136. Gary Leininger, Director of Intelligent Industrial Systems, 320 ERL, University of Missouri-Rolla, Rolla, MO 65401

137. Oscar P. Manley, Office of Basic Research and Engineering, Mathematical, and Geosciences Division, U.S. DOE, MS ER-15, Washington, DC 20545

138. Worthy N. Martin, Computer Science Department, University of Virginia, Charlottesville, VA 22904

139. Robert A. McLauchlan, Texas A&I University, Department of Civil and Mechanical Engineering, Campus Box 191, Kingsville, TX 78363

140. Alex Meystel, Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104

141. Ryszard Michalski, George Mason University, Department of CSC, Artificial Intelligence Center, 4400 University Drive, Fairfax, VA 22032

142. Christine Mitchell, Man-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332

143. Thomas Mitchell, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

144. Amit Mukerjee, Computer Science Department, Texas A&M University, College Station, TX 77843

145. J. Nievergelt, Chairman, Department of Computer Science, University of North Carolina - Chapel Hill, Department of Computer Science, Chapel Hill, NC 27514

146. Celestine Ntuen, Department of Industrial Engineering, McNair Hall, Room 405, North Carolina A&T State University, Greensboro, NC 27411

147. Eui H. Park, Department of Industrial Engineering, North Carolina A&T State University, Greensboro, NC 27411

148. Joey Parker, Department of Mechanical Engineering, College of Engineering, University of Alabama, Drawer ME, Tuscaloosa, AL 35487-2998

149. Frank Paul, Mechanical Engineering Department, Center for Advanced Manufacturing, 318 Riggs Hall, Clemson University, Clemson, SC 29634-0921

150. Richard Paul, Department of Mechanical Engineering and Computer Science, University of Pennsylvania, Philadelphia, PA 19104

151. J. J. Persensky, Division of Licensee Performance and QA, Human Factors Assessment Branch, Office of Nuclear Reactor Regulation, US NRC, Washington, DC 20555

152. Larry Peterson, U.S. Army Human Engineering Lab, Building 459, Attn: SLCHE-CS (L. Peterson), Aberdeen Proving Ground, MD 21005-5001

153. Harry E. Pople, Decision Systems Laboratory, University of Pittsburgh, School of Medicine, Pittsburgh, PA 15261

154. Michael J. Rabins, Department of Mechanical Engineering, Texas A&M University, 100 Engineering Physics Building, College Station, TX 77843-3123

155. Raj Reddy, Director, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

156. Karl N. Reid, College of Engineering, Architecture, and Technology, Oklahoma State University, Stillwater, OK 74074

157. John Roach, Virginia Polytechnic Institute and State University, Department of Computer Science, 562 McBryde Hall, Blacksburg, VA 24061

158. Bernie Rock, Office of Technology, Support Programs, U.S. Department of Energy, Washington, DC 20545

159. William Rouse, Center for Man-Machine Systems Research, Georgia Tech, 225 North Ave., NW, Atlanta, GA 30332

160. T. G. Ryan, 5546 Falmead Road, Fairfax, VA 22032

161. George Saridis, Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, 15th Street, Troy, NY 12180

162. Paul Schenker, Technical Manager, NASA/JPL Space Telerobotics Program, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

163. Thomas Sheridan, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

164. Ching-Long Shih, Center for Robotics and Manufacturing Systems, University of Kentucky, Breckinridge Hall, Lexington, KY 40506-0056

165. Wes Snyder, North Carolina State University, Center for Community and Signal Processing, Box 7914, Raleigh, NC 27695-7914

166. A. Swain, 712 Sundown Place, SE, Albuquerque, NM 87108

167. James S. Tulenko, 202 Nuclear Science Center, The University of Florida, Gainesville, FL 32601

168-181. Harold P. Van Cott, National Academy of Sciences, 2101 Constitution Ave., NW, Washington, DC 20418

182. Richard A. Volz, University of Michigan, Director of Robotics Systems Division, Department of Electrical Engineering and Computer Science, Ann Arbor, MI 48109

183. C. Wickens, Aviation Research Laboratory, University of Illinois at Urbana-Champaign, Champaign, IL 61820

184-185. Yuan F. Zheng, Department of ECE, Clemson University, Clemson, SC 29634

186. Robotic Systems Lab, Department of Industrial Engineering, North Carolina A&T State University, Greensboro, NC 27411

187-196. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831

197. Carl Steidly, Department of Computer Science, Central Washington University, Ellensburg, WV 98926

198. J. M. Giron Sierra, Santa Hortensia, 41, 1, 28002, Madrid, Spain