310

Number 310, December 1986
ISSN 0097-8515

TRANSPORTATION RESEARCH CIRCULAR

Transportation Research Board, National Research Council, 2101 Constitution Avenue, N.W., Washington, D.C. 20418

ARTIFICIAL INTELLIGENCE AND THE AIR TRAFFIC CONTROL SYSTEM

mode
4 air transportation

subject areas
12 planning
21 facilities design
51 transportation safety
54 operations and traffic control
55 traffic flow, capacity, and measurements

Report of the Workshop on Artificial Intelligence held at the National Academy of Sciences, Washington, D.C., October 23, 1985

Report Edited by Herbert J. Guth, Consultant

CONTENTS

ACKNOWLEDGMENT

David J. Sheftel, Chairman, TRB Committee on Airfield and Airspace Capacity and Delay

The organization and conduct of this workshop on a subject of such depth as expert systems and the possible application of this technology to air traffic control automation has been a task of challenging proportions. As Chairman of the sponsoring Transportation Research Board Committee, I am especially aware of the amount of time and personal dedication James P. Loomis contributed to make this workshop the success which it was. I express my gratitude to Jim and to Battelle Columbus Laboratories for making him available for this important work.

INTRODUCTION AND SUMMARY

James P. Loomis
Battelle Columbus Laboratories

This is the report of the workshop on artificial intelligence and air traffic control conducted by the Transportation Research Board Committee on Airfield and Airspace Capacity and Delay, A1J05, October 23, 1985, at the National Academy of Sciences, Washington, D.C. Participation in the workshop was by invitation only and was limited to those who were considered expert in the field.

The purpose of the workshop was to assemble key national figures involved with the development of the air traffic control system and/or with artificial intelligence (focus on expert systems) research and development with the intent of better defining how artificial intelligence (AI) may be incorporated into the air traffic control (ATC) system of the future. As far as is known, this was the first formal meeting dealing with this particular AI application. It is useful, and of some interest, to describe the circumstances leading up to this workshop.

The Committee on Airfield and Airspace Capacity and Delay is one of several TRB committees focused on air transportation concerns. It is chaired by David J. Sheftel, former Director of Research and Development for the Federal Aviation Administration. In 1984, as part of its renewal, the committee created several subcommittees through which it would conduct its activities. One of these, chaired by James P. Loomis of Battelle, was given the charter to focus on future ATC technology. Joining Loomis in this endeavor were Henry Lum of the National Aeronautics and Space Administration (NASA) Ames Research Center, Amedeo R. Odoni of the Massachusetts Institute of Technology, and Alfred C. Robinson of Battelle Columbus Laboratories.

In exploring work already under way, the subcommittee recognized that other bodies were already assessing requirements/opportunities in several technology areas. A case in point was the work of the Radio Technical Commission for Aeronautics, Special Committee 155, which was dealing with future navigation, communications, and surveillance technology. After some deliberation, artificial intelligence was selected for emphasis. The subcommittee believed that the work in this field held great potential for the future ATC system, and that, by contrast, little visibility was being given to this area's possible contributions to future ATC needs.

As its contribution to the AI/ATC arena, the subcommittee decided to sponsor a closed workshop where experts in the field could report on the status of work under way and exchange ideas on areas of opportunity deserving increased emphasis. It planned for the issuance of a proceedings as the means for disseminating the fruits of the workshop.

The AI work under way, mostly on expert systems, which was directly related to ATC, was fairly limited and was, not surprisingly, being funded by the Federal Aviation Administration (FAA). This work was viewed as an important centerpiece for the workshop. At the same time, however, related expert-systems work was seen as important. This work was under way at such places as the NASA Johnson Space Center and the Department of Defense Advanced Research Projects Agency (DARPA) and was included in the workshop.

Six presentations were made at the workshop on various aspects of the problem. These were followed by dividing the attendees into two small groups for discussion. Finally, the groups joined together for a plenary session. The remainder of these proceedings reflects, in substantial detail, what transpired during the workshop. They also include invited comments from the participants.

The report is organized as follows: (1) Introduction and Summary; (2) Presentations; (3) Group Discussions; and (4) Individual Inputs. A list of participants is included at the end of this circular.

Six presentations were made to set forth what is taking place in AI in other areas which has applicability to ATC. These were not intended to be all inclusive but do represent an interesting and varied sampling and were intended to provide a common point of departure for the subsequent group discussions.

The speakers were not asked to prepare papers in connection with their presentations. Rather, they were asked to make liberal use of visual aids. This was done with the objective of enhancing the quality of the presentations and providing graphic material for the proceedings.

To provide coverage of each presentation in the proceedings, the plan was to combine the graphics for each talk with its corresponding part of the workshop transcript. That is basically what was done for each of the six presentations which follow. While editing their part of the transcript, some authors made extensive changes. Others incorporated their graphics by reference. The result is that, in some cases, the final product looks very much like a formal paper.

The small group discussion summary reports provide the reader with the most concise summary of the conclusions reached by the workshop participants. Among other results, this workshop reaffirmed some of AI's more important attributes -- in any arena, such as ATC, it can provide much needed speed to decision making, capture valuable expertise, facilitate training, and improve all parties' understanding of a system and its operation. Participants highlighted some functional areas where AI might usefully be employed in the not-too-distant future. These areas include flight plan generation, real-time conflict resolution, severe weather detection, and flow control (traffic management). Various AI research needs were emphasized. Finally, some guidance was set forth for selecting and carrying out AI demonstrations in the ATC system.

PRESENTATIONS

FAA AIR TRAFFIC CONTROL DIRECTIONS

Paul J. Neumann, Federal Aviation Administration

This presentation describes the Federal Aviation Administration's (FAA) Advanced Automation Program (AAP). Emphasis is on the program element referred to as Automated En Route Air Traffic Control (AERA), which touches on a few possible areas for future research in artificial intelligence (AI). The Advanced Automation System replaces the National Airspace System (NAS) Stage A with enhancements such as sector suite, functional enhancements and AERA 1. This will provide increased reliability, maintainability and availability; increased ability for the system to predict air traffic control (ATC) operational problems; and increased ability to accommodate future hardware/software/human operations (HW/SW) system capabilities.

FAA has a commitment to the nation, Congress, and industry to improve safety, improve productivity of air traffic controllers, and reduce maintenance and user costs. AERA has the objectives of allowing users to fly direct, fuel efficient routes, increasing controller productivity, and reducing operational errors.

Advanced Automation Program

The FAA has divided the AAP into several individual elements in order to ease the transition and realize benefits on an evolutionary basis during the National Airspace System modernization. To go from today's system to a fully automated system in one step would be difficult and unwise. The first phase is the rehosting of the existing software on modern computer hardware. The next phase is to replace the existing plan view displays with a multiple display environment, which FAA refers to as a sector suite along with the development of a new hardware and software system. This system will include the first capabilities of AERA. The objective of this first phase is to remove existing operational constraints that limit the number of direct routes that can be safely granted. It has been estimated that this will result in a 3 percent fuel saving to air carrier and commuter operators. FAA refers to this first automation enhancement as AERA 1.

The next planned automation enhancement, AERA 2, has the objective of increasing controller productivity while maintaining all the advantages of the first phase. The FAA is in the process of developing a functional specification for this phase and plans to add this requirement to the AAS development when it is sufficiently well described so that the contractor can develop the software.

There is a final phase where all of the capabilities are tied together and the AAS will automatically generate and deliver to controlled aircraft conflict free, fuel efficient clearances. As this presentation describes the AERA capabilities in greater detail, the potential for application of AI techniques will become clearer. Figure 1 describes the AERA objectives.

AERA Objectives

AERA 1 is now incorporated in the AAS specification and both of the design competition phase contractors are incorporating it into their designs. It was added to the contract in two pieces. AERA 1 is primarily an early detector of potential problems and tools that can help the controller in solving the identified problems. Problem identification specified by air traffic includes:

- Violation of safe separation distance
- Incursion into restricted airspace
- High sector workload.

AERA 1 has been generated in two parts:

- AERA 1 AAS Base Specification
- AERA 1 AAS Modification.

AERA 1 AAS modification was generated based on current operators' position that the base AERA 1 would not meet AERA 1's stated goal. However, the base AERA 1, if developed as a partial package, would increase workload with no payoff to airspace users. Therefore, AERA 1 is being developed in its entirety to achieve FAA acceptance.

The first AERA 1 was composed of a 4-dimensional trajectory estimator, an aircraft-to-aircraft conflict detector, an aircraft-to-airspace conflict detector, and a sector workload estimator.

Figure 1. AERA objectives.

AERA 1

- TO INCREASE THE NUMBER OF USER PREFERRED ROUTES GRANTED RESULTING IN MORE EFFICIENT USE OF THE AIRSPACE AND GREATER FUEL ECONOMY FOR USERS OF THE SYSTEM

AERA 2

- TO INCREASE CONTROLLER PRODUCTIVITY WHILE MAINTAINING OR ENHANCING SERVICES UNDER INCREASING DEMAND. AUTOMATION AIDS AND ADVICE WILL BE SUBJECT TO CONTROLLER APPROVAL.

AERA 3

- TO FURTHER INCREASE PRODUCTIVITY BY AUTOMATICALLY PERFORMING MANY CONTROLLER FUNCTIONS. SOME OF THE ACTIONS WILL BE UNDERTAKEN WITHOUT CONTROLLER INTERVENTION.

The 4-D trajectory estimator will project an aircraft's 3-dimensional position in time along the planned path that the pilot has filed. This capability is the foundation on which all the other FAA functions are built. It will make these projections for all controlled aircraft.

The aircraft-to-aircraft conflict probe will use these projections to determine when aircrafts' paths are predicted to cross within a specified distance and time. The accuracy with which it will predict position will determine the look ahead time that will be useful to display the predicted conflict to controllers. The present estimate is that this will be about 20 minutes. Naturally, the goals of minimizing false and missed alarms will conflict with the look ahead time. The engineering and operational tradeoffs will be made during the operational testing.

Similarly, the aircraft-to-airsapce conflict probe will look for predicted aircraft incursions into military operations areas and other types of restricted airspace. This look ahead capability could conceivably be for the entire flight since the restricted areas are precisely defined. The need for this capability becomes more important as more aircraft fly on the structured route system and controllers will be less able to look at a direct route and determine if there is going to be an incursion several sectors away.

Finally, the original AERA 1 sector workload factor will predict sector workload for each active sector. It will look at, as a minimum, traffic pattern complexity, predicted communications, number of aircraft in the sector, and number of climbing and descending aircraft. Both the raw data and an index will be provided to the controller and to supervisors.

Because the basic objective of AERA 1 was to increase the number of fuel efficient routings, an FAA Air Traffic Service sponsored AERA advisory group was established to review the AERA 1 functions in light of the stated objective. This group consisted of 6 en route and 4 terminal area active controllers who had the charter to conduct the review and make recommendations to FAA management on any changes necessary in the basic concept needed to achieve the objective. This team spent approximately 12 weeks reviewing the existing documentation, questioning the design and system engineers, reviewing AAS documents, and examining the implications that each of the proposed tools would have on their operations and how they might be applied in the AAS/sector suite environment. This review resulted in the recommendation that 6 additional functional capabilities be added to the AAS/AERA 1 system. After a review in headquarters, the AAS contract was modified to include essential parts of the group's recommendations and these are now considered to be integral parts of the first implementation of the AAS.

These additional capabilities are:

- Trial plan processing - Provides a controller with the capability to construct temporary flight plans that can be tested for problems by other automation capabilities,
- Conformance monitor - Periodically compare the trajectory of an aircraft with its track position to ensure that positions agree within parameter tolerances in the lateral, vertical and longitudinal direction,
- Reconformance aid - This function will create a trial plan that provides assistance to the controller for re-establishing vertical or lateral conformance between track and trajectory position,
- Detection of flow restriction violations - This function will detect and flag to controllers violations of local flow, central flow, metering, and airport restrictions,
- Controller reminders - Controllers will be alerted to planned changes in altitude with restriction notices, and expect further clearance information,
- Limited resolution aid - This capability will generate up to four trial plans when a problem is identified by other AREA tools and the controller requests help. The maneuver menu will include altitude change, lateral route offset, speed change, and vectors.

AERA 2

Next, the AERA 2 functional capabilities will be discussed. The objective of AERA 2 is to increase controller productivity. Current projections indicate that the volume of aircraft operations will increase substantially over the next decade. Studies and actual experience have shown that there is a maximum number of aircraft that an ATC specialist can safely control for a particular operational situation. Thus to handle this anticipated increase in traffic either the number of controllers must be increased or the control environment must provide more automation tools to reduce the workload per aircraft. Increasing the number of controllers and sectors quickly yields small marginal improvements in system capacity due to increased controller-to-controller communications and coordination. The FAA has decided that the best way is to increase the level of automation in order to allow ATC specialists to handle more aircraft and has bundled these automation tools into a program called AERA 2.

AERA 2 has the goal of providing automated problem resolution and clearance coordination and will allow the use of data link for clearance delivery. The problems that have been identified for automated resolution are aircraft-to-aircraft, aircraft-to-airspace, and flow restriction problems. The generation of these resolutions is complicated by the requirement that each of the resolutions be cross-correlated to assure that resolution of one problem does not cause another problem. In addition the resolutions should be acceptable and understandable to controllers and operationally feasible and fuel efficient for the system users.

Considering the mix of aircraft, the various operational environments, the total system demand and the localized high-use airport demands, reaction times of pilots and controllers in various workload environments, hardware and software capabilities and limitations on the ground and in the air, designing software that satisfies these complex and sometimes conflicting conditions will not prove to be easy. In fact, it will be a great challenge to build this system.

There are also several enhancements to the AERA 1 functions that will become operational concurrent with the implementation of AERA 2. These are:

- Enhanced situation monitor - This adds the capability to identify aircraft which were previously denied a route or altitude request due to airspace or flow restrictions when the restriction is changed or eliminated.

- Enhanced conformance monitor - Adds a capability to notify a controller when an aircraft -- still in conformance -- appears to be headed for a violation of the conformance region,
- Enhanced trial planning - Adds a capability for the automatic re-evaluation of a trial plan at time intervals specified by the controller,
- Enhanced controller reminders - Reminders are extended to provide a more complete list of reminders for the controller.

Reviewing the above functions it becomes clear that the core and critical element of AERA 2 are automatic problem resolution (APR). A plan is being laid out to demonstrate the feasibility of developing an operationally useful APR function. This has been one of the concerns expressed by staff members of the General Accounting Office. They believe that it is important to demonstrate feasibility and productivity gains from implementing AERA functions. A method being looked at for this demonstration is the use of expert systems and rapid prototyping to build in incremental steps towards a full AERA 2 demonstration capability.

AERA 3

The final phase, AERA 3, has the goal of automatically generating conflict free, fuel efficient clearances to pilots without controller intervention. This program is in the early stages of research and functionally is undefined. Yet, it appears that AI could be used to advantage in automating many of the cognitive tasks that controllers perform today.

Finally, there are two important areas for future research:

(1) Development of an AI base simulation capability to use as an AERA evaluation tool;

(2) Degree of resolution intelligence that can be developed and used to solve ATC problems.

In conclusion, there is a real need to examine the proposed ATC system enhancements relative to the real and anticipated AI technologies to help FAA plan for their incorporation into our future systems.

DISCUSSION

Question When you say "Limited Resolution" do you mean that the resolution is limited or the aid is limited?

Paul J. Neumann Both - the way the system is currently envisioned there would be a limited number of resolution strategies resident in the machine and, if no resolution was identified, the controller would be notified of the failure of the machine to find a resolution. The controller could not expect to find a machine generated solution to every problem. In addition the resolutions might not supply a resolution that was problem free for the probe time horizon. For example, a conflict could be resolved but the resolution would cause a future flow problem.

However, the limited resolution aid provides an opportunity for the introduction of artificial intelligence into the ATC system. It should be noted that this is an opinion not shared by all in the FAA. It appears that techniques now employed in expert systems would yield resolutions that would be more acceptable to controllers and pilots than a strictly mathematical solution of trajectory equations and conflict avoidance maneuver strategies. Introduction of AI technology into a tool that was not necessary for operational safety, since the traffic in this environment would be essentially the same as in the pre-AAS environment, would allow the evaluation of AI performance in the ATC environment and provide a knowledge base for the development of the more sophisticated AERA 2 resolution strategies.

Question You have been talking primarily about things related directly to controller productivity. In reducing costs there are other avenues for reducing operational costs and increasing controller productivity. I am thinking primarily of the maintenance area where AI could have an application and system reconfiguration when you have hardware failures. Do you have any idea of how relative operational costs of control versus these other support capabilities stack up?

Paul J. Neumann The primary expense to FAA is salaries of both controller and maintenance personnel. Maintenance costs are being controlled through the replacement of existing equipment with highly reliable solid state equipment, including the current 9020 computers with IBM host computers. The application of AI to maintenance of electronic equipment was being looked at by Dr. Siewierek of Carnegie Mellon University. The FAA was jointly funding research on building an expert diagnostician but unfortunately the program ran out of funding. Steve Alvania from AES-320 may speak about that later.

The major personnel costs to the FAA are incurred in the Air Traffic Service. FAA looks at gains in controller productivity as being relatively more important because of the larger cost saving if the same percentage is applied to that side of operations.

Question The user is troubled today because of severe flow control restrictions. In AERA 1 it seems that you are planning considerable research in automation. Why are flow restrictions, metering, and airport restrictions included?

Paul J. Neumann FAA is looking for more flexibility in the AERA 1 system that will minimize reliance on imposing restrictions. However, considering the projected growth in traffic, it seems to be prudent to plan for some restrictions in capability limited and impacted areas.

Question It seems rather than expediting traffic through the use of AREA 1, you are continuing to plan for the imposition of restrictions.

Paul J. Neumann At the moment no one envisions that flow control will go away in the near future. FAA cannot control weather events which may reduce capacity or close airports. Nor can FAA control the demand that users put on airports at specific arrival and departure times. Someone here may be smart enough to come up with a way to devise an AI system that devises a solution.

DARPA PILOT'S ASSOCIATE PROGRAM

John P. Retelle, Jr.
Defense Advanced Research Projects Agency,
U.S. Department of Defense

Michael Kaul, BDM Corporation

The pilot's associate program (Figure 1) is one of three applications in the Defense Advanced Research Projects Agency (DARPA)'s strategic computing program. DARPA is an organization within the Office of the Secretary of Defense that looks at high-risk new technology. It was formed in 1958 by President Eisenhower basically in response to the Sputnik surprise. The Department of Defense wanted to reduce technical risk and essentially protect the nation from technical surprise. DARPA examines high-risk, high-payoff projects, works closely with the military services, and hopes to transition these new technologies that are shown through demonstrations to the services. For financial guidelines, DARPA tries to fund to accept the risk. That means essentially that DARPA puts up the "up-front" money to create a demonstration and then looks forward to passing the technology to the services.

Organization of DARPA

DARPA is organized into several offices. The Tactical Technology Office (TTO), in the Air Warfare Division deals with both air warfare and land warfare. The Defense Sciences Office deals with more basic research, fundamental G.1 level research. The Information Processing Techniques Office deals with new computers, super computers and the strategic computing program. There is also a small activity in directed energy. Most of that work has gone to the Strategic Defense Initiative Office. DARPA also has a Strategic Technology Office that deals with innovative strategic activity, and, also, with satellite assets.

The DARPA budget is significant, and it permits much flexibility to start new technologies. It is a lean organization. Out of 162 people, only 36 are program managers. This has made it possible to keep the organization small, to start the programs very quickly and to stop programs very quickly as well. DARPA typically uses service agents. There is no procurement activity within the agency which is really tied to the use of agents, not only for technical support, but, also, for procurement and program support as well.

Pilot's Associate Program

The Pilot's Associate Program is an application to try to help the pilot of single-seat fighter aircraft.

Figure 2 shows the overall strategic computing program at DARPA. DARPA has been working in artificial intelligence for about 15 to 20 years. This particular initiative was meant to expand the technology base and to focus those activities into some applications. Each one of these boxes is a separate program with a program manager. The infrastructure provides much of the computer power. The technology-based activity has three functional areas. The first one deals with the chip level technology, new advances in silicon, in VLSI and, also, in gallium arsenide. DARPA is finding that a particular chip allows us extraordinary computing speed and, also, is impervious to disturbances from nuclear events, etc.

The hardware area deals essentially with processors that are meant to exploit the symbolic processing necessary for artificial intelligence; thus, much work is parallel processing. About a dozen different parallel processing architectures are being assessed. DARPA intends to get the speed and computing power needed through parallel processing. There is some consideration now of 1 million processors being connected together at one time. Some of the early examples have 128 node list processors, such as the BBN Butterfly machine.

The Pilot's Associate Program has a different focus (Figure 3). It puts artificial intelligence in the cockpit to help the human pilot. It has the use of expert systems and of speech interfaces, but it has the added difficulties of the power, space

Figure 1. Pilot's Associate Program.

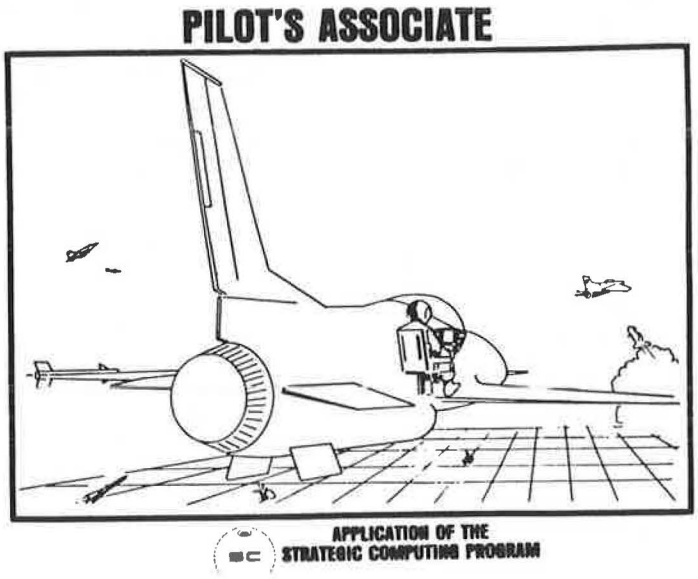


Figure 2. Overall Strategic Computing Program at DARPA.

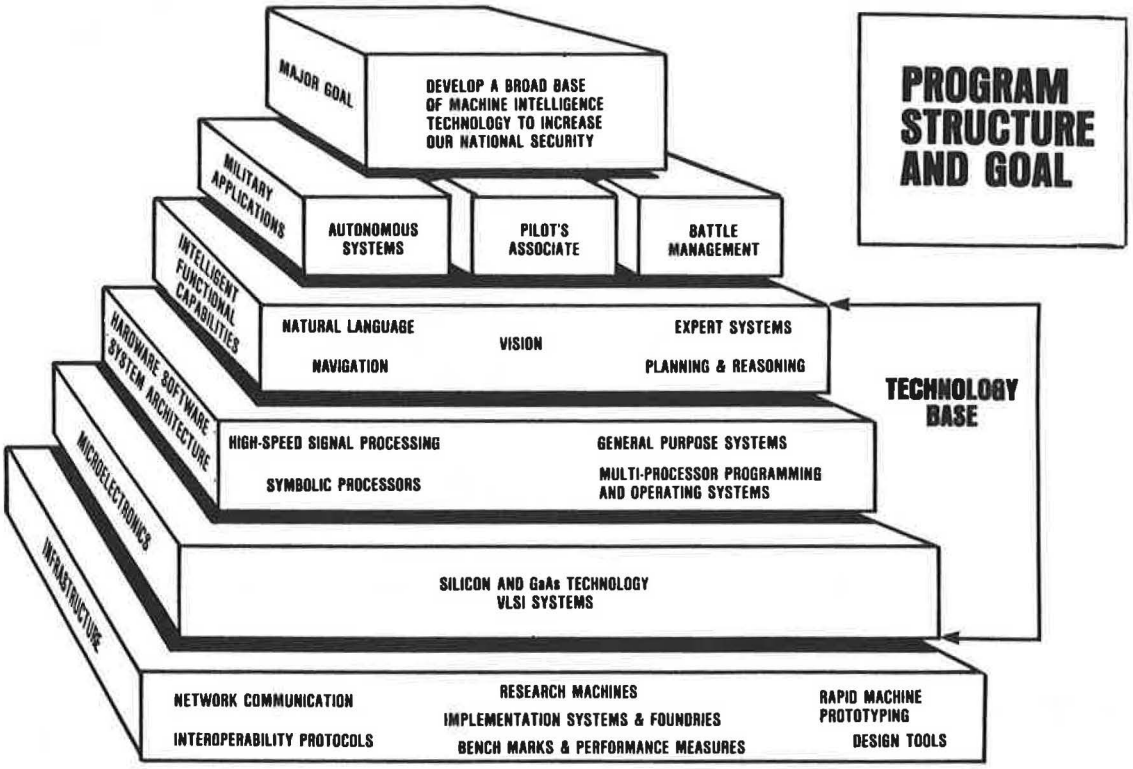
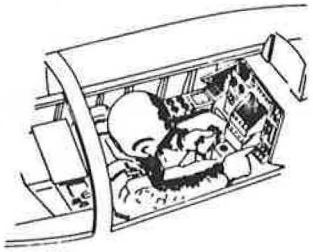


Figure 3. Pilot's Associate Program Overview.



PROGRAM OBJECTIVES

- EXPLOIT DARPA STRATEGIC COMPUTING TECHNOLOGY BASE
  - EXPERT SYSTEMS
  - PARALLEL PROCESSING
  - SPEECH SYSTEMS
  - PILOT-VEHICLE INTERFACE
- REAL-TIME INFORMATION MANAGEMENT
- PROVIDE DECISION AIDING IN DEMANDING HIGH THREAT SCENARIOS

FUNCTIONAL AREAS

- SITUATION ASSESSMENT
- SYSTEM STATUS
- MISSION PLANNING
- TACTICS ADVISORY

DEVELOPMENT ENVIRONMENT

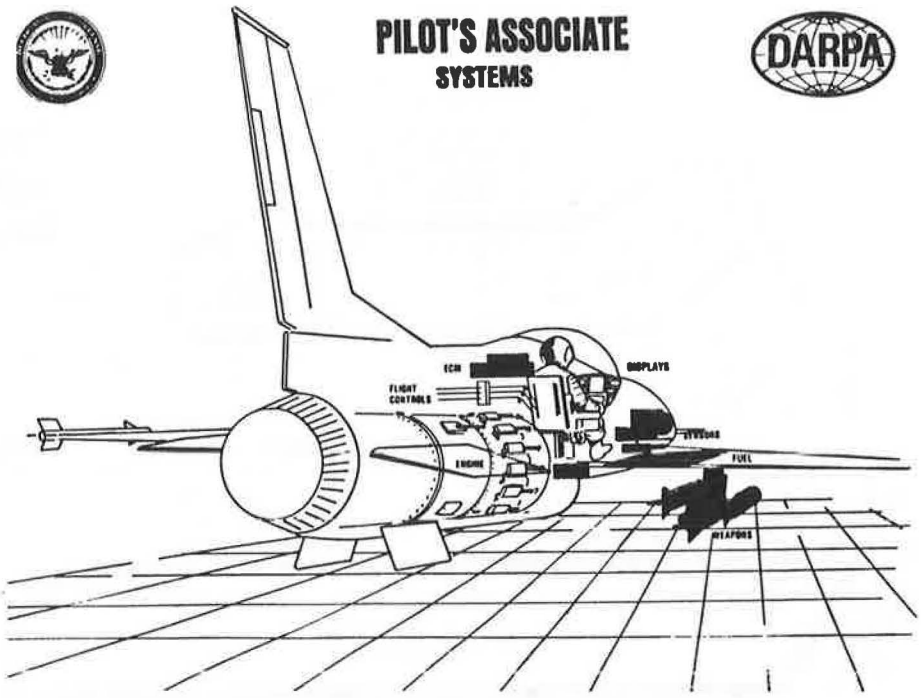
- COMPUTER LABORATORY — PART TASK SIMULATORS
- FULL MISSION SIMULATORS

8543/8SW/GW28/1

PROGRAM MILESTONES

FY86-88:	PHASE I — KNOWLEDGE ENGINEERING, SYSTEM DEVELOPMENT
FY86:	DEMO ONE — FEASIBILITY AND IMPACT ASSESSMENT
FY87:	DEMO TWO — NON-REAL-TIME PART TASK SYSTEM SIMULATION
FY88:	DEMO THREE — NEAR-REAL-TIME TOTAL SYSTEM SIMULATION
FY89-90:	PHASE II — INTEGRATION OF TECHNOLOGY BASE PRODUCTS
FY90:	DEMO FOUR — FULL MISSION REAL-TIME SIMULATION, TOTAL SYSTEM EVALUATION

Figure 4. Pilot's Associate Systems.



and real time computing requirements of an airborne processor and, also, the very intimate interface with a human being.

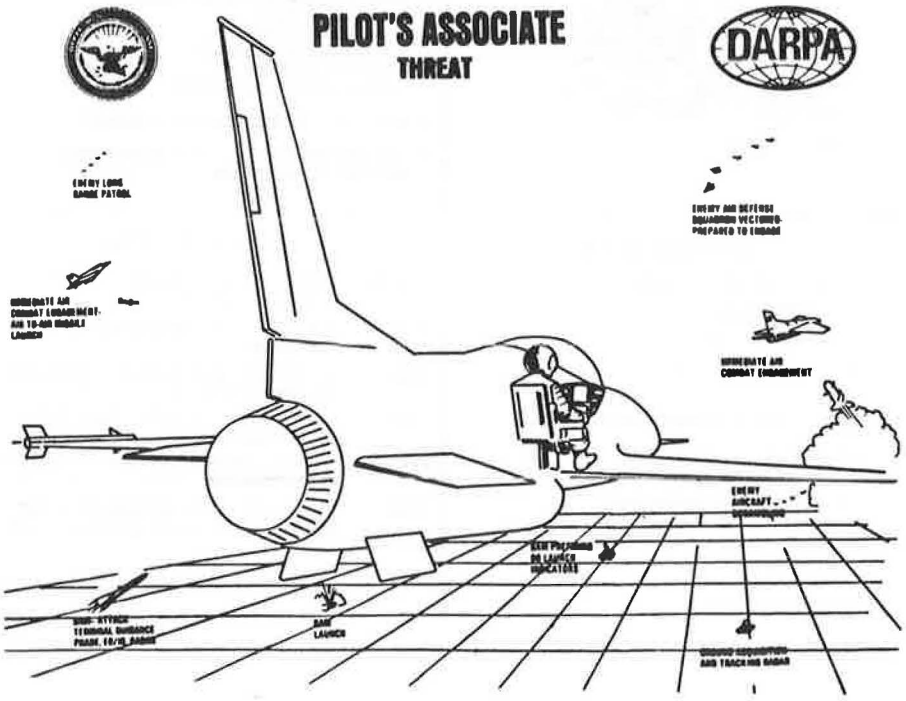
The areas of AI that will be exploited from the strategic computing program are cooperating expert systems, parallel processing for avionics, speech systems and new ideas in pilot vehicle interface.

The two things needed for the combat fighter pilot are to help deal with managing the vast amount of information he has available to him and provide decision aiding for very tough mission scenarios.

Functional Areas

This presentation will now go through the functional areas, the program development and the program milestones in the following charts. The first functional area has to do with aircraft systems (Figure 4).

Figure 5. Pilot's Associate Threat.



Basically, much information is available from on-board sensors. The pilot has many instruments that he looks at and tries to make assessments of the status of his systems. He can certainly understand the balance provided by the airplane manual, the Dash-1 in the airplane, but it is often difficult for him to assess trends and predictions that may affect his mission later on.

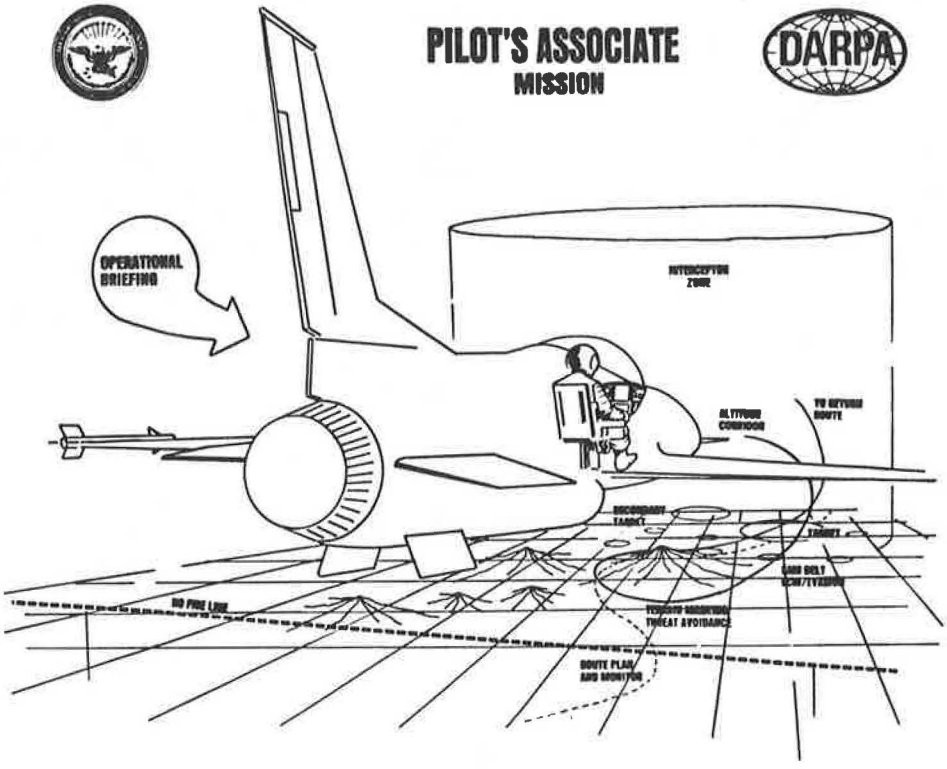
One of the things that the systems status monitor draws on is the existing knowledge in expert systems technology in terms of diagnostic kinds of systems, whereas this is a single-seat fighter pilot type application. The ability to diagnose vast amounts of inputs, both in the cockpit and out of the cockpit would have significance in commercial application as well.

The obvious ones, of course, are aircraft emergencies. Obviously it brings into mind events such as the DC-10 crash in Chicago. In that case, if the pilot could have had the information to make the right decision at the right time, perhaps he could have done something about it. Or, perhaps a large amount of processing could cause a semi-autonomous action to take place which would help the pilot in those non-book types of situations that do tend to occur and tend to characterize most aircraft emergencies. The ability to diagnose large amounts of information is significant in air traffic control. Earlier there was discussion about degradation of systems and the ability to recognize faults, incipient software failures in the air traffic control systems. These are the kinds of things that expert systems do well, and have been proven to do well, particularly in existing commercial applications such as mycin which is a diagnostic system. The idea is that the systems status monitor would be able to recognize faults, predict faults, help the controller with aircraft that have misread the altitude directions, frequencies and so forth and look ahead to problems rather than catch them when they actually occur.

The other functional area has to do with threat (Figure 5). Fighter pilots call this situation awareness. How can they get the information from sensors to say, "Who is out here; what is their perceived intention, and can a rank ordered list be made of who is going to kill me first?" It is a very difficult computation not only to determine who is out there, but to predict what they are going to do in the future (Figure 5).

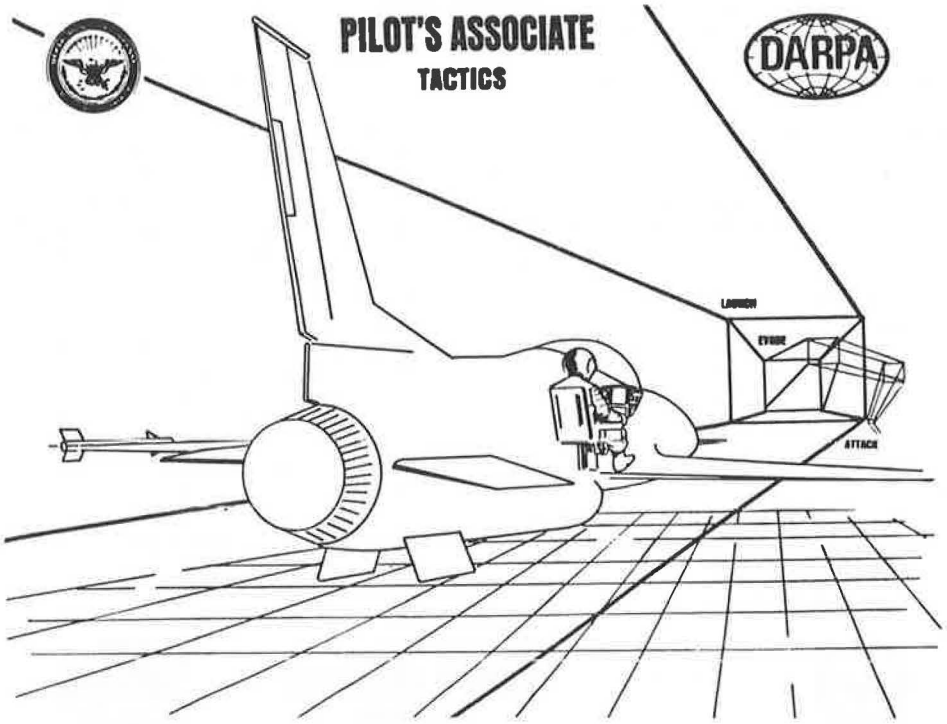
Again, transitioning in terms of what this particular expert system can do in commercial application, expert systems in this particular application would focus on handling vast amounts of data, going through large data bases and using heuristic search techniques to come up with the right answer quickly. Earlier the discussion focused on user preference routes and things such as conformance and trajectory prediction and so forth. This type of expert would help you with that kind of situation where the expert system could take into account things such as where the flow of traffic is, what the weather is today, what the pilot preference is in terms of efficient routing, fuel status, separation criteria and some sort of constraint relaxation in case you have problems such as flow control. Constraint relaxation is the kind of thing that expert systems are best at.

Figure 6. Pilot's Associate Mission.



Military pilots spend much time preplanning their missions (Figure 6). Not only do they have to worry about their own aircraft, other aircraft on the flight, or an entire battle, they also have to worry about going through routes and having proper identification, etc. That is difficult, and to keep all that information sorted out and available to the pilot in real time is also difficult, but the harder problem is when you get rerouted. If you are 200 feet at night, IR base with the lantern system in an F16 and AWACS calls you and says, "Don't go down this valley, but go over here," you cannot plan that mission on a knee board. You are

Figure 7. Pilot's Associate Tactics.

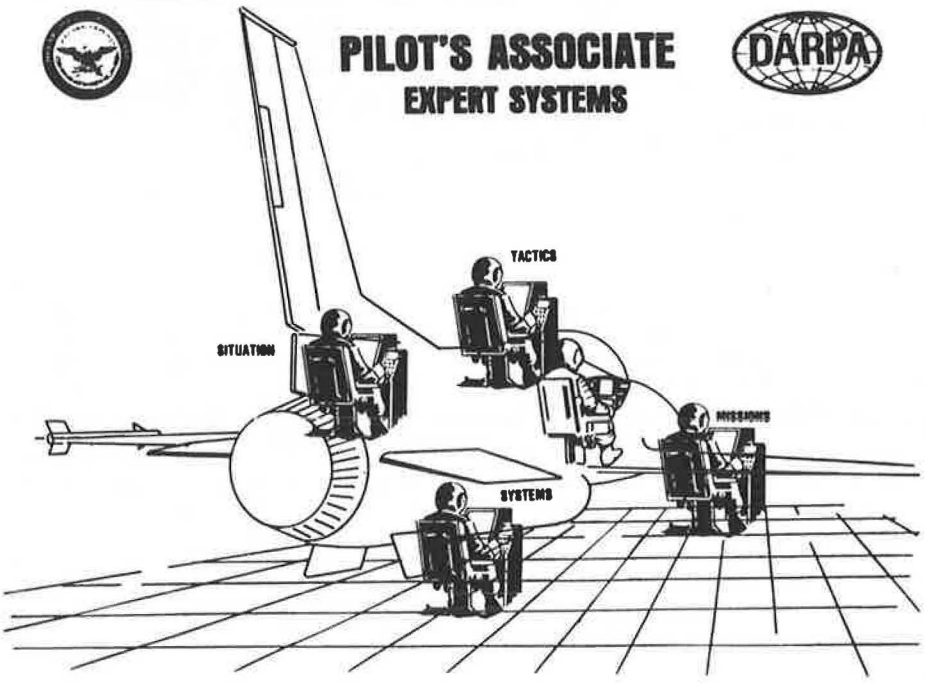


going to need some help, and that is the type of replanning that we intend to do in this particular system.

Thus, expert systems in this case are drawing on the ability to monitor, plan and replan. Again, expert systems are known for performing those kinds of activities well. Going back to the notion of planning and replanning, consider a high-density route, such as the Atlantic routes from Washington to Miami that are bounded on both sides by restricted areas. The ability to conform to restrictions while, also, taking into account changing weather situations, pilot desires and turbulence that has cropped up as a result of weather, is a way of taking into account what the current state ought to be and using an expert system for real time replanning as the situation dictates. In other words, a resolution aid, a conformance aid, and trajectory prediction. Again, expert systems help when rules need to be relaxed. This could be a way to relax the constraints caused by flow control in real time on a high-density route.

The last functioning area has to do with tactics (Figure 7). There is need for a rapid solution, perhaps almost semi-automated or automated

Figure 8. Pilot's Associate Expert Systems.



to help the pilot out. The question is: "Should I attack or should I evade? Can I consider my situation? Does the adversary see me or not? Should I go for him? Should I just get out of the way? Am I locked on to by two incoming missiles. Which one should I avoid first? What is the preferred tactic? When should I perform that tactic?" Real time operations are performed that are probably partially or fully automated.

The tactics expert system tends to draw on the AI facility to predict and optimize and to do that quickly with much information. The example here in terms of air traffic control is a high density terminal area, a situation where there are a number of runways, parallel runways, multiple operations and thunderstorms moving across one of the approach paths. The tactic then would be to aid in figuring out as far back as possible which aircraft need to be slowed down, how the air traffic controller can help the pilot manage his own energy status and, also, to consider thunderstorm cell movement, where the microbursts are, where the shears are and to try to form an immediate set of tactics around the air field.

Thus, the model being created for the military audience is a collection of expert systems that assist the pilot, a surrogate crew of sorts (Figure 8). These are not "R2-D2" crew members. They are expert systems that have rules that are based upon human crew members, such as a copilot or an electronic warfare officer, a flight engineer or the pilot himself. It is meant to assist him rather than act in his place.

In the case of the civilian area, there may be multiple crew members, rather than a single fighter pilot, and each crew member may have his own expert system available to him. Then you have a communications challenge between the crew members themselves and between the individual expert systems.

This summarizes what these four cooperating systems may do in the four functional areas, monitoring information and comparing it with the models that you have. A functional construct is shown in Figure 9. Four expert systems are linked together. The input data on the left side includes data from planning, data from sensors external to the airplane and internal sensors as well.

A capable pilot interface then interfaces with the pilot and, also, with the various control systems on the aircraft. Now, the AI researcher looks at it and says, "There are some problems here." One is the notion of cooperating expert systems and needs some work. Where does the data base reside? Is it separated in the four expert systems? How is the control of information between those expert systems managed, and who gets the data file? You can have a situation where you have an engine fire; you are being painted by a number of diverse radars, and you do not know who they are and what their intent is. You have been asked by AWACS to go some place else, and you have two missiles locked on to launch on your airplane. Who gets to the pilot first? What is the most compelling disaster? It certainly is a research and development challenge.

Program Phases

The program that was estalbished basically has two phases (Figure 10). The first phase of the pilot associate is a laboratory exercise and runs for three years, through 1988, where it is intended to develop expert systems, to link them together, and build the basis for some preliminary laboratory evaluations.

The second phase runs from 1989 to 1990, and two things happen in this phase. For the first activity, we transition into full mission simulators. Very capable simulators exist that can create not only the aircraft environment but also, the whole mission battle environment as well. It is believed that the pilot's workload level can be increased enough so that some additional value from the pilot's associate is apparent.

The second thing that happens in this stage is that the contractor participants will be requested to rehost their software from the preliminary machines into a parallel processing environment of their choice from the DARPA Strategic Computing Program. The first phase, Phase 1 is under competition right now. A number of proposals have

Figure 9. Pilot's Associate Functional Construct.

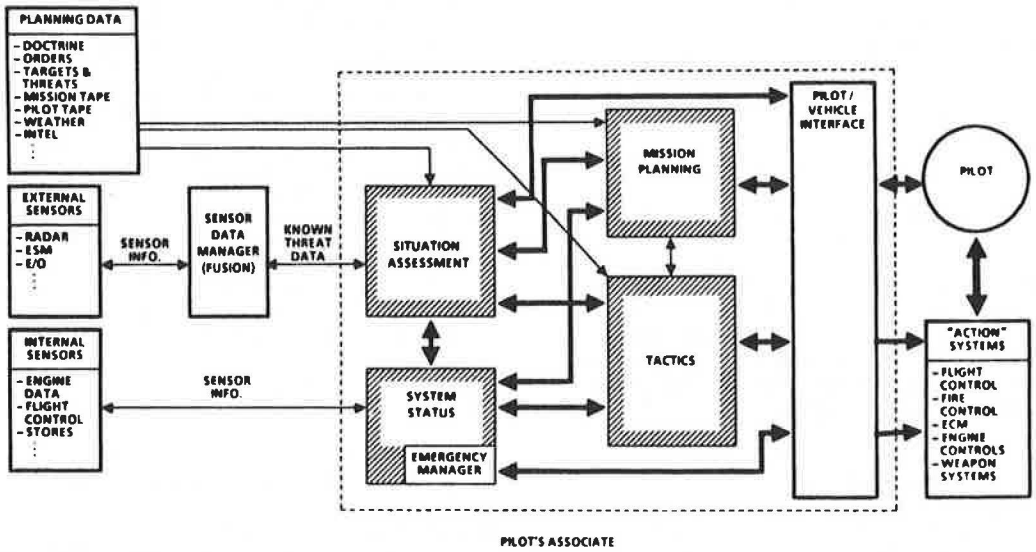
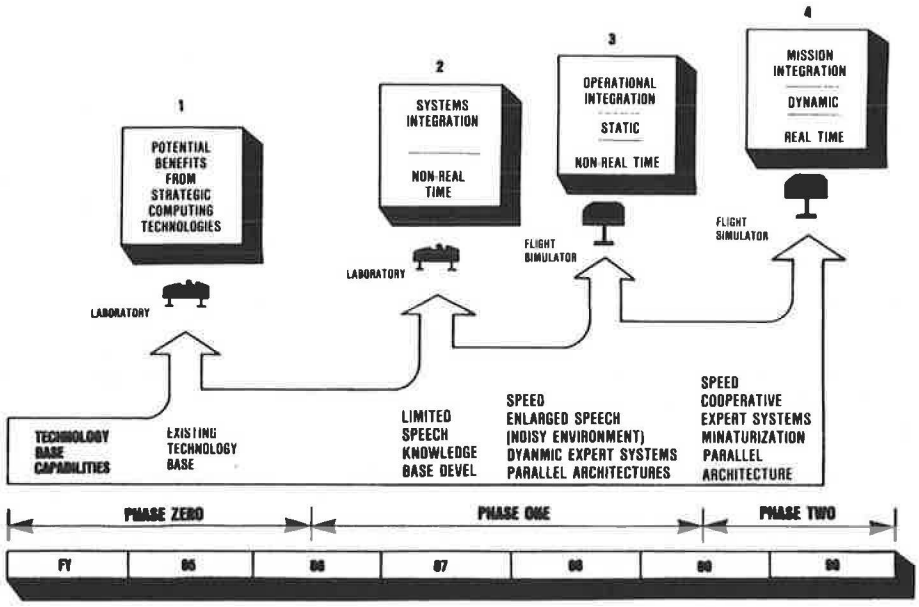


Figure 10. Pilot's Associate Phases and Demonstrations.



been received. The types of people that responded to our RFP were teams of companies. That might be a major airframer, an avionics organization and an AI organization.

Figure 10 shows the first stage which includes the first three demonstrations and the second phase of the program. Also, throughout the program, products will be infused from the DARPA strategic computing technology base, as well. Contractors in the program expect to benefit from that.

The areas from the strategic computing technology base, G.1 level, basic research, expert systems, architectures, especially parallel processing, and speech will be customized into new generation systems that can be used for the pilot's associate application.

The operational impact of this system is first, that the vast amount of information from communications and sensors that is coming and presently available for fighter pilots can be used. There is now just too much information in too short a time and the pilot can hardly use it all. There is also some false and ambiguous information from the other side. Second, many pilots do not have combat experience, and there is some concern about their "buck fever" in the first days of the conflict. It is expected that the expertise contained in the expert systems can help them through that problem area. In many cases there was a need for assurance of capability against a large threat.

In summary, DARPA intends to address some very difficult tactical missions for single-seat fighters. Much information is available and there are very difficult threat situations. The aircraft themselves are very capable. What DARPA intends to use from the strategic computing program are new ideas in expert systems and parallel processing. New avionics and new displays are needed and, also, interfaces with current avionics and use of speech and natural language processing.

There should be survivability and mission effectiveness improvements through the simulator demonstrations. In short, the basic objective of the program is to help the pilot be the best fighter pilot he possibly can be.

DISCUSSION

Question Who is working the software for the processes you mentioned earlier?

John P. Retelle, Jr. It is being worked by many of the contractors in the DARPA program. Some are working hardware. Some are working software. The software activity, also, happens in the technology base. These are the disciplines that one normally associates with artificial intelligence: expert systems, natural language, computer vision, etc. It is hoped to provide new generation systems in each of these technical areas and then to transition them to the applications. There are three applications programs. The first one is an autonomous land vehicle. Basically it is a vehicle that exploits expert systems and computer vision to find its way through a very difficult terrain. The program has been under way for a while. It uses a particular set of computer architectures. It has already started its operation and had its first successful demonstration test some months ago. In the battle management area, there are two programs. One has to do with air-land battle management, where we try to organize the information available to a battlefield commander. Another area has to do with helping the commander of a carrier battle group. One part is the CINCPACFLEET command center in Hawaii, and the other one is the battle command center on the USS Vinson aircraft carrier. Much work is being done with very large rule based expert systems with the natural language interface with those systems and speech understanding as well.

Question Regarding the Vinson, have they published any documents on this program.

John P. Retelle, Jr. No, they just had their first preliminary demonstration. This is kind of a real time program. It is ongoing right now, and rather than having published things, what has been used is user group working groups that get together and pass on the information. Of course, it is the intent to publish the results, both positive and

negative, that we have out of this. It has not happened yet.

Question Why is there the need to have four separate expert systems? Has it to do with speed of the hardware or the conceptual difficulty in building these things or is it associated with somehow selling this technique to pilots and explaining it to them in that way?

John P. Retelle, Jr. It is basically our partitioning of the functions that the pilot or air crew does. A rather lengthy exercise looking at the functions that fighter pilots perform was done, and they were partitioned into different functional areas. Then, based upon the computing power that was necessary for each one of those functional areas, it was decided to break them up, rather than do it in one massive system.

Estimates were made of the processor sizes and computational power available for each one of the systems, and it appeared that breaking them up was the best way to get something that was reasonable to transition in the time frame of the program.

Michael Kaul There is another side to that. In addition to mapping the expert systems function in terms of the functionality of the pilot himself, there was a desire to try to map those functions in terms of what expert systems had virtually proven themselves to be best at.

John P. Retelle, Jr. There is a programmatic consideration as well. Some of them are easier than others. For the systems status manager, for example, people think they already know how to do that. However, the tactical planner requires very rapid prediction capability. Therefore, a range of technical challenges was desired.

Question Can you see those things coming together in the future and does that imply less need for a pilot's active involvement?

John P. Retelle, Jr. You asked two questions. The first is can they come together in the future? It depends upon the computing capability that comes out of the technology base, and it, also, depends upon what is learned from this program. It may be found that even though they could come together, it makes no sense to do so. The second question you asked is will this replace the pilot? It is not the intent of the program to replace the pilot. It is the intent to assist the pilot and to step up to the very difficult behavioral question of how does a human being deal with this new computing capability. The other two strategic computing programs really do not step up to that. These are challenges in this program.

Question A question on the new pilots with a low level of experience. What is your feeling regarding a pilot with low experience who comes to rely on the automation. Would the result be not that he is gaining experience up to this point, as much as a degradation of your good pilots down to the lower level of the automation.

John P. Retelle, Jr. That is a difficult question. The approach, the philosophy that we had was to help pilots with combat situations. They are expected to be fully trained, to be combat ready, but also, to react in some personal, internal way to the first shot that gets fired at them. That is the type of situation to be dealt with, such as your controller dealing with his first life or death emergency. That is the type of situation, rather than to supplant a training activity. That is not intended.

Michael Kaul There are situations that the controller is operating in daily. You get thunderstorms, and you have to draw on every skill you have ever known; if they have not been practicing there are no skills. They are gone. They get periodic training, but these are fully qualified controllers. But after a while you begin to rely on the machine, and you really have not been doing this.

John P. Retelle, Jr. It is a very difficult question, how much you rely on the machine or how much you allow your operators to rely on the machine. There is no answer to that yet. It depends really upon the success with these new types of AI-based systems. If the crews or the controllers or pilots feel they can rely on the machinery, maybe there will be some of that, but it is going to take a while.

Question You mentioned cooperating expert systems. Is that the idea of two different expert systems or do you mean a partial, say 90 percent, automation of the function?

John P. Retelle, Jr. Do you mean in the expert system cooperating together, sharing the information that they have in different knowledge bases, resolving conflicts and advice that they both come up with to present to the pilot? As far as the total automation, that is still the prerogative of the pilot. We intend to allow the pilot to choose to accept or reject the advice that is offered by the pilot's associate.

Question You mentioned using a parallel processing architecture. Is that a constraint because you are dealing with an airplane, and you would use something different if you had a land-based system such as the air traffic control system?

John P. Retelle, Jr. The reason for going to parallel processors is that we expect through parallelism to have an enormous increase, 10^4 level increase, in computer speed. That is the reason for doing it. Now, I don't mean to rehost everything in avionic processing into a LISP or into an AI or parallel processing environment. It will be done where it is appropriate. What we are going to see in the pilot's associate is a hybrid collection of predictive processing, such as the dynamic programming, the normal, numeric processing and perhaps a LISP-based program or some other AI language base.

Comment One of the concerns of the controllers in an automated environment is the interface between the machine and the actions that they want to take. They said that if they were just updating constantly what happened in the real world into the machine, keeping that up to date, that was going to be a problem. Eventually the natural language processing interface can help ease some of that dialogue with speech recognition.

John P. Retelle, Jr. The interface with the human pilot is a tough one and I do not want to have a program that does copy and redesign. What I do want to look for is new ideas for the cognitive part, if you will, between the human and the computer -- an example in speech is the speech work they have done at NASA Ames, in helicopters which has indicated that if the speech is too frequent,

the pilot tends to think of it as a person. If your computer on a CRT gives you wrong answers, you say, "Ah, the computer is mixed up." If the speech system gives you the wrong answer, the pilot says, "It lied to me." So, you have to be very careful about the use of that interface.

Question Is there interaction between your program and the FAA?

John P. Retelle, Jr. There is none, and that is really a hoped-for outcome of this meeting.

SHUTTLE AIR TRAFFIC CONTROL EXPERT SYSTEM

Robert H. Brown and C. J. Culbert,
National Aeronautics and Space
Administration, Johnson Space Center

This presentation will examine some of the work being done at the Johnson Space Center, particularly in the application of expert systems to a number of problems. Some 10 or 12 expert systems have either been developed or are in the process of being developed. A number of them probably apply to the subject of air traffic control appropriateness.

An expert system called a controller has been built and basically what it does is evaluate the status of the hardware and software at Mission Control Center in real time and then advise the computer supervisor as to the status. This replaces two people who had very tedious jobs but, also, very, very important jobs. In the process of implementation is what is called computer controllers; they will actually control all the antennae and do antenna management during real time in the space transportation system. It also includes an interactive graphics capability for the controller interface. Schedulers have been developed which schedule both people and resources in very specific narrow areas. These are now in operation and have proved to be much more capable than the people who were doing the job. In process of development is an automated rendezvous and docking expert system which would include three expert systems.

It has also been found that as some of these expert systems are built they are excellent trainers. There is also ongoing research with natural language and speech recognition, neither of which are very satisfactory at this time.

This presentation will now describe an application of expert system technology to a typical mission control center monitoring problem. The mission Planning and Analysis Division currently works for many shuttle support activities. One of these is high-speed ground navigation. Currently teams of three people work during the ascent and entry phases of missions from the space shuttle. These people work on a standard console consisting of five CRT devices, five digital display driver panels, one computer terminal or manned equipment device and one DRK panel, which is a pushbutton device.

These operators monitor and control the processors that work during the ascent and entry phases of the mission. The first one is the high-speed trajectory determinator or HSTD. The other is the Delta-State update processor, SUP. The configuration that is believed possible using expert system technology is to reduce the manpower from a three-person team to a single-person team aided by an expert system. Currently these people monitor and control the HSTD high-speed trajectory determinor. This processor uses data from one to three radar stations processed through a Kalman filtering technique to generate estimates of the shuttle's position and velocity. The state update processor is a program which monitors the on-board computer navigation performance and compares it to the ground navigation performance. Currently it requires two or more years to train a person to operate on this console. This is a very complex, highly detailed, monitoring problem, and there is a tremendous amount of data coming in. The operators work with a display which has over 110 parameters on it, and each of these numbers changes every 2 seconds. As such, a series of lights go on and off every 2 seconds. There are as many as 50 or 60 lights the operators must monitor. The operators will take three prime actions with regard to the filter. First, they can exclude or include data coming in from a particular radar station. They might include it if there is a good solution or good data to be incorporated into the solution or conversely, they may exclude a station to prevent bad data from being included in the filter solution. Also, they may restart the filter to prevent propagating ahead or finally if they have to, they can stop the filter to prevent a bad solution from being used by anybody else. These operators are responsible for doing whatever actions are required to maintain the health of a carbon filter and to provide the best possible estimate of the ground velocity and position of the shuttle.

The expert system was used to emulate or recreate the decision-making process of the units. The expert system was used or built using an ART, the automated reasoning tool developed by Inference Corporation. When describing an expert system the size is estimated by counting the number of rules. This expert system has about 100 rules. It was developed in less than two months using the automated reasoning tool. The performance of the expert system often called NAVEX is very impressive. It is able to monitor all the information coming in the computer's monitor at a rate that is four to eight times faster than the humans currently work at. That gives us the capability to add in more words to the expert system to make it more robust and more flexible and, also, potentially gives us the capability of increasing the rate at which we monitor information. The expert system will be improved to include the ascent case or ascent rules and bring those on line as well.

A more detailed description of the high speed ground navigation expert system which has been developed at Johnson Space Center is given in the attached appendix. In brief, the chief functions demonstrated by this project are as follows.

First, it demonstrated that expert system technology is feasible for use in NASA's console operations. This was a complex monitoring problem, requiring much expertise and human skill. The expert system was able to handle that. Also, the expert system was able to meet the real time needs of the system working at a very high data rate. Second, the expert system did not require a highly experienced or highly expensive knowledge engineer to develop it. It was developed in a very short time typically by the NASA personnel with help from the Inference Corporation.

In summary, the work had a number of objectives. One was to demonstrate the current state of the art in terms of the technology that is available now in hardware and software. The purpose was to demonstrate that this technology is being used and that expert systems can be developed rapidly at a

reasonable cost. This project was done in a few months. Typically we run well under six months on any expert system application, with one or two people working on it. It has also been possible to develop the in-house capability to do the knowledge engineering kinds of jobs, but that may be peculiar to NASA applications since many systems analysts have done this.

One of the things that an expert system does is to keep track of all the potential problems; as a problem develops the human tends to get tunnel vision and be concerned with that emergency. The expert system, of course, does not do that. It watches for all the potential problems that may come up later which is a tremendous advantage, certainly in the problems that are being examined.

The need for three console operators is that three pair of eyes are needed to look at all the data that is coming in, and decisions are a general consensus. The method used was to take a consensus of eight experts to compare that consensus with the decisions made by the expert system. In every case the expert system made the appropriate decision and in general, being conservative, made it earlier. It was also found that most of the time the expert system was idle, waiting for more information which means that basically four to eight times as much data can be added.

DISCUSSION

Question What machine was that running on?

Robert M. Brown Symbolics. But we are using every major expert system builder that is available.

If you are going to build expert systems, you are going to do it rapidly. The best thing to do is get the most power that is available to do that. As we get into developing an expert system, the experts are asked how many rules they use in making their decisions. For example, in this case one said, "Oh, 1000." Another said, "Two thousand", and one said "Several thousand." This was reduced to 110 rules, and it was found that in most cases there is a tremendous reduction in what an individual or an expert thinks he uses in making decisions and how you can implement those in terms of rules.

Today's technology can build kernel expert systems very rapidly and at a very low cost. In today's environment, there are a tremendous number of companies selling hardware and software expertise which looks good and sounds good. When you really get down to trying to do a lot of different things with it, it is just really not very powerful at all.

Another problem is verification and validation. Expert systems normally do not degrade gracefully. When you get out to the edge of the knowledge domain, they will start giving you very, very idiotic answers. So, you have to be extremely careful how you build them.

We do not know much about verification and validation. The way I am doing that is I have a fiberoptics line to the mission control center in my area, and we run in parallel with the control center until we are satisfied.

One of the things you have to be careful about is that expert systems do not exhibit imagination, originality or common sense. You know it is bad for airplanes to crash. An expert system does not. You know if you drop something it will fall, and an expert system does not know that. You know if a pilot is making a decision based on incoming information, and he sees some obstruction, he knows he needs to fly around it. Unless you tell the expert system, it does not know. So, there are some things that we have to be extremely careful about in terms of building expert systems. Our expectations are using them in consultant modes, developing them in kernel form on well-understood, well-defined problems. The technology is here today to do that.

Questions Do you have any mission-critical applications in expert systems operations?

Robert H. Brown No, this of course is the NAVEX and it will go in as a consultant. It is still in the evaluation phase.

One of the big problems is target machines. If you want to rapidly develop an expert system, the best thing to do is use a symbolic machine or LISP-type machines that are available and being developed rapidly, but then when you get ready to deploy them, it is very, very expensive.

APPENDIX

HIGH SPEED GROUND NAVIGATION EXPERT SYSTEM (NAVEX)

C. J. Culbert
NASA Johnson Space Center

Artificial Intelligence (AI) is flourishing outside the bounds of its traditional academic environment. Spurred by the success of a few key technologies, commercial development is placing more and more of the computer advances pioneered by the AI world into the applications environment. These technologies are beginning to reach maturity in a number of areas, and can make a significant controbution to all aspects of the space program.

The AI section of the Mission Planning and Analysis Division (MPAD), Johnson Space Center (JSC) is developing applications which apply AI technologies to NASA problems. In particular, the AI section is using expert systems to aid highly trained humans accomplish complex tasks. An example of this is the Navigation Expert project, NAVEX. NAVEX is an expert system built to aid in the operation of the high speed ground navigation console in the Mission Control Center (MCC) at the Johnson Space Center. This project was one of the first expert system development projects in MPAD. It was begun as a feasibility study to examine the potential for the use of state-of-the-art artificial intelligence hardware and software in typical JSC applications. The prototype expert system for NAVEX was developed by the Inference Corporation in conjunction with MPAD personnel in about three months. NAVEX was designed and built on a Symbolics computer (a specialized LISP machine) using the automated reasoning tool (ART), a product developed by Inference. ART is a sophisticated software tool used to develop an expert system. ART allows programmers to work in a very high level language with advanced programming constructs, and takes full advantage of the development environment of the Symbolics computer (1).

The console task currently requires teams of three people who monitor and control the high speed trajectory determinator (HSTD) during the ascent and entry phases of a shuttle mission. They also monitor the Delta-State Update processor (SUP). These teams work at a typical MCC console with 5 CRT displays, 5 digital display devices (colored status and warning lights), a computer terminal for data entry, and a set of push buttons for command

entry. By using an expert system advisor, manpower requirements can be reduced from three people to one person while also reducing training effort and improving response.

The HSTD is a Kalman filter program which uses data from one to three radar stations to generate an estimate of the shuttle's position and velocity. This state vector is then used by numerous other flight controllers and/or programs in the MCC which need the shuttle's current state vector. The HSTD generates a state vector every two seconds, using the current radar measurement to propagate the previous state vector forward. The HSTD also estimates the error in the radar measurements. The SUP is used to help monitor performance of the onboard navigation systems and to compute state vector updates. The processor computes the differences between the onboard state vector and the HSTD state vector.

A tremendous amount of information is presented to the controllers. The prime display has over 100 parameters which change every 2 seconds. Typically there are two or three other displays available for additional information on the other CRT's and there are between 30 and 50 blinking lights on the digital display devices which need to be monitored.

The console operators are responsible for maintaining and improving the performance of the HSTD. The operators monitor the noise and bias statistics generated by the HSTD, looking for trends or data anomalies. They can specifically include or exclude the data from a particular radar station in the Kalman processing. They can also completely restart the filter, i.e., drop all previous state vector estimates from the current solution and start fresh with the next set of radar data. Deciding what actions to take and when to take them is a process which requires a high level of human expertise. Console operators require 2 or more years of training to become experts. They need to have knowledge of how a Kalman filter works, how radars work, how a particular radar has reacted in previous missions, the potential effects of their actions on other flight controllers, how to input data and commands, and of course, the flight rules. Other responsibilities of the high speed navigators include advising other flight controllers of the current reliability of the HSTD solution, and the validity of the state vector update. A more complete description of the duties of the console operators is available in reference (2).

The techniques used to develop NAVEX are applicable to numerous other monitoring type problems. A better description of this architecture is available in Reference (3). NAVEX uses a "synchronous data acquisition architecture", i.e., data is input to the expert system at regular intervals. The output from the HSTD is presented to NAVEX for reasoning every two seconds. NAVEX also makes use of the ART viewpoint architecture for temporal modeling. This modeling does not include a complete history of the past (although it could). Instead, information of importance in each cycle is saved as state information to a special viewpoint. At the beginning of each clock cycle a new viewpoint is created and the current HSTD information is asserted into it. NAVEX then reasons about this information, together with state information. The use of state information allows NAVEX to reason about trends in the data. In addition to making recommendations, NAVEX can note trends developing in the data and can both alert the users and set a watch in the state information.

NAVEX operates in four phases during each clock cycle. The first phase advances the clock, creates the new viewpoint and calls the LISP functions which gather the data. The second phase reasons about the current data to generate all possible actions. Multiple recommendations are allowed to coexist during this phase (within the current viewpoint). During the second phase, all recommendations are considered independently. Any previously noted trends or watches for possible problems are considered and updated in this phase. The third phase then considers interdependencies among all the independent recommendations and refines the set of possible recommendations. The fourth phase selects the best of the recommended actions for execution. Finally, a new clock cycle is begun. A complete description of the NAVEX system architecture is available in reference (4).

The high speed console was chosen for the NAVEX project because it was typical of shuttle monitoring type problems, it represented a good test of a complex operation, and also was a well defined problem. The expert system had to demonstrate not only expert level reasoning capability, but also had to meet the real-time data needs. It had to be capable of monitoring large amounts of data at high rates and make critical recommendations in real time. To date tests of the NAVEX system have shown it capable of meeting these stringent requirements while still functioning at an expert competence level.

The NAVEX project demonstrated two key points. First, that current AI technology was capable of handling typical JSC type problems, both in complexity and speed. Second, the prototype system was developed in a very short time without relying on specialized "knowledge engineers". These points open the door to numerous other uses within NASA, both for the space shuttle and the space station. The AI Section is developing other systems which further demonstrate the use of AI technology in NASA applications.

References

1. ART Reference Manual, Inference Corporation, 5300 W. Century Blvd., Los Angeles, CA 90045.

2. High-Speed Ground Navigation Console Procedures - STS Operations, NASA Technical Memorandum 82-FM-46, NASA/Johnson Space Center, Houston, TX 77058.

3. M. C. Maletz and C. J. Culbert, Monitoring Real-Time Navigation Processes Using the Automated Reasoning Tool (ART), Proceedings of the Aerospace Applications of Artificial Intelligence Conference, September 1985.

4. M. C. Maletz, NAVEX System Architecture and User's Manual, Inference Corporation, 5300 W. Century Blvd., Los Angeles, CA 90045.

POTENTIAL USE OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN AIR TRAFFIC CONTROL

Antonio L. Elias and John D. Pararas
Massachusetts Institute of Technology

A Blasted Introduction to Artificial Intelligence

Edward Feigenbaum, in his "Handbook of Artificial Intelligence" defines artificial intelligence (AI) as "the part of computer science concerned with designing intelligent computer systems". This is a very easy definition to make, since it shifts the burden of definition to another one, namely that of "intelligent computer system". Feigenbaum then proceeds to define intelligent computer systems as those which "exhibit the characteristics we associate with intelligence in human behavior". Unfortunately, this one is not very helpful either, since we now must define intelligence itself, a rather formidable task.

But even if we were able to define human intelligence, we would still have a problem, since "intelligent behavior", when applied to computers, cannot be equated with intelligent behavior in general. As little as a hundred years ago, computing the square root of a number was unequivocally a manifestation of human intelligence, since it required a number of decision-making steps depending on the signs of intermediate values, remainders, and so on. Yet today, nobody would call the square root calculating ability of a computer intelligent behavior.

In view of this inability to satisfactorily define intelligent behavior as applied to a computer, some people have slightly altered the classical definition to mean doing with a computer something you normally don't expect a computer to be able to do. This definition appears to be satisfactory, since taking square roots, for example, is something you expect a computer to do, so a system that takes square roots of numbers is definitely not an artificial intelligence system, while a system that composes concert music does appear intelligent, since computers do not usually compose concert music.

The problem with this new definition is that it is self-defeating: the moment one builds a computer system that does something you do not expect a computer to do, it does it, so it ceases to become an artificial intelligence system. As paradoxical as this may seem, it actually makes some sense: today, one can purchase battery-powered toys at a department store that do a better job at synthesizing speech or playing chess than the most sophisticated experimental equipment did just ten years ago.

In view of this difficulty in defining what precisely artificial intelligence is, perhaps we should simply describe a little of its history and some of its typical products. AI research has traditionally had three distinct objectives: first, to understand the high-level workings of the human brain by constructing functional computer models of human activities, such as vision and reasoning; second, to build computers based on the brain model; and third, to build a mechanical man, perhaps by combining the results of the other two efforts.

The motivation of the first line of research is a better understanding of the human brain, and any computer functionality that may result from this research is purely secondary. The motivation of the second, to build computer systems - both hardware and software - patterned after the human brain, is the goal of building better computer systems, independently of the specific applications.

Now the third goal is the most elusive one; humans had the dream of building mechanical replicas of themselves for at least as long as they dreamed of flying. If we have been able to fulfill the dream of flying, is there any reason we will not be able to build a true robot? Perhaps, but we should be very careful not to identify artificial intelligence with only this goal. If we do, we will miss what perhaps are the most useful benefits from AI research. However, these benefits are not the intelligent machines per se, but rather the computer technology that has been developed as a consequence of the quest for machine intelligence. AI is also not only expert systems. The popularity and press coverage that expert systems have recently received have caused a lot of people to believe that the only useful product of artificial intelligence research consists of expert systems.

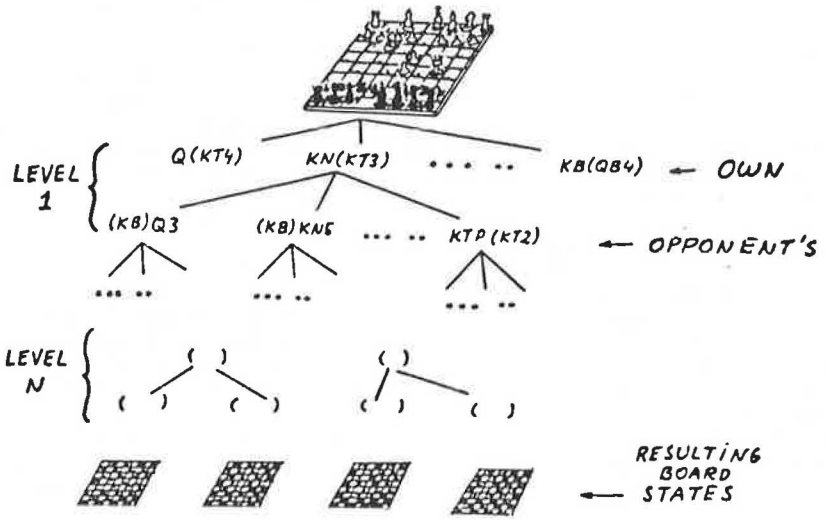
Historically artificial intelligence had its roots in the discipline of mathematical logic, sometimes also called symbolic logic, the study of the processes by means of which we construct the mental models we call mathematics. It was with the discovery, by Turing and others, that these symbols could be manipulated and operated upon mechanically with the same ease as numbers - although with a different set of operations, naturally - that the possibility of a computer performing these intelligent functions was first postulated. Indeed, Lisp, now considered to be the programming *lingua franca* of the AI community, can be considered either a programming language, or a convenient, elegant, and powerful method of expressing mathematical concepts.

The first attempts at using computers to manipulate symbols for a purpose started by defining a simple problem to be solved. The kinds of problems that early AI systems were capable of handling had two common characteristics: the goal, or problem, was very simple to state, but the solution to the problem was complex and non-trivial. The measure of success used in the development of these game-playing systems was this: could the program play a better game than the people that build it?

The common technique used in these systems was the generation of large sets of alternatives, followed by a process of search (for a desired solution), usually coupled with procedures that reduce the number of alternatives to be evaluated to a reasonable subset. In a chess-playing program, the alternatives are the sequence of legal moves and counter-moves that can be made by the program and its opponent from the current state of the board, alternatives that can be structured as a tree; the search consists in the successive evaluation of each branch of the tree to find the most convenient immediate move, evaluation that may include not only the eventual end state of the board at the end of that branch of the tree, but the likelihood of each of the opponent's moves (Figure 1).

Before these solutions can be generated, searched, reduced and evaluated, some symbols and operations must be defined; in other words, a representation of the problem must be designed. For example, a chess-playing system may operate on descriptions of the state of the chess board, that is, the position of each of the pieces; the operations that can be performed on these descriptions would include valid piece movements, or functions that measure the desirability of having a piece in a certain position relative to other pieces. Other symbols that may be involved could include standard

Figure 1. Demonstration of alternative moves on a chess board.



moves, such as the classical chess opening moves, in such a way that the system can easily recognize when the opponent has performed such a move, and know what the consequences of that move are without elaborate analysis.

In spite of the spectacular performances that such systems exhibit - few human chess players can outperform the best chess playing programs today - these efforts were in a way disappointing because of the extremely narrow focus of the results. While some of the searching and problem reduction techniques developed as a consequence of that research are applicable to a large class of problems, the problem representation aspects were extremely case-dependent: the symbols and operations developed to solve chess moves are of little or no value outside that specific problem domain.

This frustration led in the late 1960s and early 1970s to a flurry of efforts to find more universal problem representations; ideas such as problem solving systems and logic reasoning systems seemed attainable at the time. At one time work actually began on a general purpose problem solver system, with no clear limitations on what kind of problems it could solve. When it became apparent that finding truly domain-independent means of representing problems was a little too difficult, researchers then directed their efforts to more restricted, but still relatively generic problems, for example proving mathematical theorems or automatic computer programming.

Also at this time, and perhaps influenced by the success of the early game-playing programs, some individuals began to make exaggerated claims about the practical possibilities of AI systems. Actually, this had already happened before, even before the term artificial intelligence had been coined. Grossly unrealistic estimates of the potential of computers to perform intelligent functions - whatever those may be - were common during the early days of electronic computers, as exemplified in the contemporary label electronic brain.

Figure 2 was produced directly from the terminal screen in the Flight Transportation Laboratory of the Massachusetts Institute of Technology computer complex. Lines with numbers beginning with the letter C are the authors' inputs, while lines identified with the letter D are the outputs from M.I.T.'s MACSYMA program. We begin this example by typing in an equation in a form which looks very much like FORTRAN. Notice, however, that MACSYMA displays the input back in a form that resembles the way one would write this equation on a blackboard; this is possible because the representation of that equation used in MACSYMA has concepts such as power and denominator.

We then ask MACSYMA to "solve" for the variable Y in that equation; what happens next is that MACSYMA recognizes a binomial equation pattern in that expression, and invokes the rules to solve such equations that we all learned in high school. If, on the other hand, we ask MACSYMA to integrate the right-hand side of that equation, it will recognize a polynomial pattern, and invoke the classical polynomial integration rules. While we could do that ourselves, we would be hard pressed to integrate the expression shown in the next figure, which involves knowing some rather exotic rules of integration. We use MACSYMA frequently in our work, especially to manipulate rotation matrices that transform, for example, position vectors in radar site coordinates to mosaic-relative coordinates. This kind of system which is composed of an internal representation of a domain, a set of rules representing knowledge in that domain, and a set of commands that allow the user to invoke the appropriate rule without knowing the details of that rule is called a knowledge-based system, or more precisely, a stored-knowledge system;

Figure 2. A typical MACSYMA session.

```
(C1) x = 3 * y^2 - 2 * y + 17;
                                  2
(D1)                     X = 3 Y  - 2 Y + 17

(C2) solve(d1,y);
                        SQRT(3 X - 50) - 1      SQRT(3 X - 50) + 1
(D2)          [Y = - ------------------, Y = ------------------]
                              3                       3

(C3) integrate(part(c1,2),y);
                                3    2
(D3)                           Y  - Y  + 17 Y
```

Figure 3. A typical MACSYMA session (continued).

```
(C4) sqrt((x+1)/(x-1));
                                  X + 1
(D4)                          SQRT(-----)
                                  X - 1

(C5) integrate(d4,x);
                    X + 1                X + 1              X + 1
              LOG(SQRT(-----) + 1)  LOG(SQRT(-----) - 1)  SQRT(-----)
                    X - 1                X - 1              X - 1
(D5)  - 4 (- ------------------- + ------------------- - ------------)
                    4                    4              2 (X + 1)
                                                        --------- - 2
                                                          X - 1

(C6) ratsimp(d5);
                  X + 1                X + 1                  X + 1
(D6)  LOG(SQRT(-----) + 1) - LOG(SQRT(-----) - 1) + (X - 1) SQRT(-----)
                  X - 1                X - 1                  X - 1
```

the computer program has the knowledge, but does not know when and how to apply it unless specifically instructed by the user (Figure 3).

The indisputable success of these stored-knowledge systems, coupled with the failure of totally generic problem-solving or thinking programs, resulted in the expert system concept. Like the stored knowledge system, an expert system operates with symbols and operations representing knowledge in a particular field, and sets of pre-stored rules which embody knowledge, just like the integration rules of MACSYMA. The expert system, however, has two additional ingredients: first, the capability of chaining the given rules, perhaps with the help of intermediate results, to reach conclusions that are not covered by any single rule; and second, logic to direct both the invocation of the rules and the chaining of simple rules to achieve a specific objective.

Perhaps the best way to understand the notion of expert systems is to observe one in operation, The next figures will show a typical conversation with perhaps the most famous, or successful, of all expert systems: the mycin bacterial infection diagnostic program. In the mycin system a moderately large set of rules (about one hundred) is used to store knowledge about bacterial infections. But whereas in MACSYMA the rules were scanned to see if one of them satisfied the request typed by the user (e.g., integrate a given expression), mycin rules are automatically activated in a complex way by a logic in the program called the inference engine, whose built-in goal is fixed: to determine the best antibiotic treatment for a patient whose infection is not precisely known.

Figure 4. A typical mycin session.

```
---------- PATIENT-1 ----------

1) Patient's name: (first-last)
*** Fred Smith

2) Sex
*** Male

3) Age:
*** 55

4) Have you been able to obtain positive cultures from a site
at which Fred Smith has an infection?
*** Yes

---------- INFECTION-1 ----------

5) What is the infection?
** Primary-bacteremia
```

Figure 5. A typical mycin session (continued).

```
6) Please give the date and approximate time when signs or
symptoms of the primary-bacteremia (infection-1) first appeared
(mo/da/yr)
*** May 5 1980

The most recent positive culture associated with the
primary-bacteremia (infection-1) will be referred to as:

---------- CULTURE-1 ----------

7) From what site was the specimen for culture-1 taken?
*** Blood

8) Please give the date and time when this blood culture
(culture-1) was obtained. (mo/da/yr)
*** May 9 1980

The first significant organism from this blood culture
(culture-1) will be referred to as:
```

To understand why mycin behaves the way it does, indeed to understand the importance of the development of mycin, we have to review briefly what kind of problem it is solving. There are thousands of bacteria that can cause infections in humans, and hundreds of available antibiotic drugs that act specifically on a bacteria, or groups of bacteria, while having more or less desirable side effects. If the identity of the bacteria causing a patient's infection were to be known, a very precise antibiotic treatment could be prescribed. Unfortunately full identification of a bacteria from a culture requires from a few days to a few weeks, while antibiotic treatment must begin immediately. Thus, treatment is begun with only sketchy data on what kind of bacteria is involved - usually a two to three hour culture yielding only very basic information about the bacteria, not its precise identity.

Figure 6. A typical mycin session (continued).

```
---------- ORGANISM-1 ----------

9) Enter the identity of organism-1:
*** unknown

10) Is organism-1 a rod or coccus:
*** rod

11) The gram stain of organism-1:
*** gramneg

12) Have there been positive cultures from sites which are NOT
associated with infections about which you are seeking advice?
*** no

13) Did organism-1 grow aerobically?
*** yes (0.8)

14) Did organism-1 grow anaerobically
*** yes
```

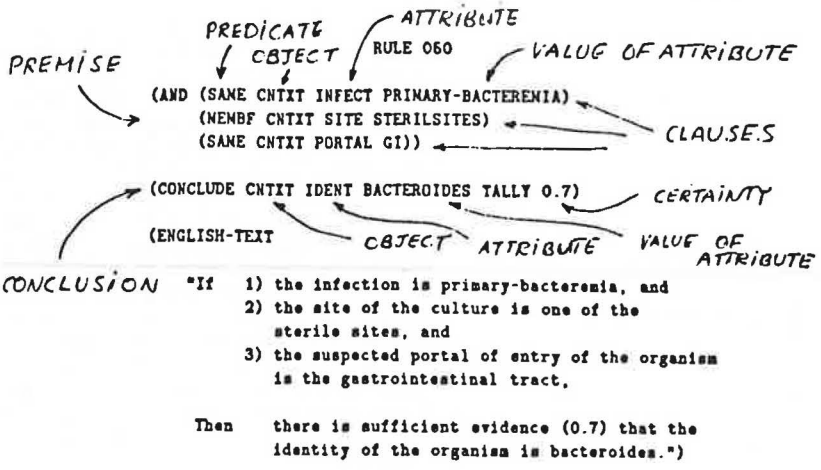
Figure 7. A typical mycin session (continued).

```
15) What is the suspected portal of entry of organism-1 into
this sterile site?
*** GI (0.6)
                . . . . . .

My therapy recommendation will be based on the following
probable infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY-BACTEREMIA
<Item 1> The identity of organism-1 may be PSEUDOMONAS-AERUGINOSA
<Item 2> The identity of organism-1 may be KLEBSIELLA-PNEUMONIAE
<Item 3> The identity of organism-1 may be E.COLI
<Item 4> The identity of organism-1 may be BACTEROIDES-FRAGILIS
<Item 5> The identity of organism-1 may be ENTEROBACTER
<Item 6> The identity of organism-1 may be PROTEUS-NON-MIRABILIS

46) On a scale of 0 to 4, where higher numbers indicate
increasing severity, how would you rate Fred Smith's degree
of sickness?
*** 3
```

Figure 8. A typical mycin session (continued).

```
47) Does Fred Smith have a clinically significant allergic
reaction to any antimicrobial agent?
*** no

48) Fred Smith's weight in kilograms (or <number> pounds):
*** 70

[Rec 1] My preferred therapy recommendation is as follows:

In order to cover for Items <1 2 3 5 6>:
     Give: GENTAMICIN
     Dose: 119 mg (1.7 mg/kg) q8h IV [or IM] for 10 days
           Comments: Modify dose in renal failure

In order to cover for Item <4>
     Give: CLINDAMYCIN
     Dose: 595 mg (8.5 mg/kg) q6h IV [or IM] for 14 days
     Comments: If diarrhea or other GI symptoms develop,
           patient should be evaluated for possible
           pseudomembranous colitis.
```

Figure 9. A typical mycin rule.



The way mycin achieves this objective is by assuming a very large number of possible alternatives and then asking the user to provide information that it can use to eliminate as many alternatives as possible, until all the information is exhausted. This technique is known as backwards-chaining the rules.

The conversation shown in Figures 4 to 9 is a little long, but interesting. Mycin's questions are preceded by a number, while the user's answers are preceded by three asterisks. After the usual basic questions about the patient, mycin checks in question (4) that the basic operating premise, that is, the existence of an infection, is indeed true. If one were to answer no to that question mycin would simply say goodbye.

At the very beginning of the conversation mycin printed the label PATIENT-1; after question (4) it prints the label INFECTION-1; these labels are an indication of the context of the conversation. When humans exchange information verbally we implicitly establish a context in which indefinite articles such as it or the have a unique meaning. Although mycin does not understand English it always has a current context, or implicit object of inquiry which begins with the patient, switches to the first infection (for that patient), and then may change to an organism, to a culture, change back to the patient, and so on.

After establishing that the type of infection is known, so that a series of questions leading to the identification of the type or possible types of infection is not necessary, mycin then proceeds to find out what laboratory information has been obtained on the organism or organisms producing the infection. Answers to a question, including the answer "don't know", dynamically modify the sequence of successive questions. Note also that the user's answers can be followed by a number in parenthesis, such as in question 13; this indicates the degree of confidence that the user has in that piece of information, with 1 indicating absolute certainty, and 0 being equivalent to a don't know answer.

After about forty or so questions mycin is ready to display a conclusion; perhaps it is satisfied that this conclusion has a low enough uncertainty factor, or, more likely, the user has begun to answer "I don't know" to so many questions that mycin decided that to give up asking. In any case mycin displays first, its conclusions regarding the possible identity of the organism causing the infection. As you can see, it is not a single conclusion, but rather six conclusions. Next, after three additional questions mycin proceeds to issue a 'preferred treatment', preferred in that there may be other treatments covering the same set of bacterial infections and which may be preferable to the user for reasons that mycin cannot handle (for example, local availability).

The next figure shows the form of a typical mycin rule. On the top of the figure is the text of the 50th rule, as stored in mycin, while a comment in English at the bottom of the figure explains the meaning of the rule (for the benefit of humans). The rule has two parts: a premise and a conclusion. If the premise is true then the conclusion is true, much like an *if - then* statement in a traditional programming language such as Fortran. A program using this kind of rules is sometimes called a production system.

The premise is in itself composed of the boolean, or logical combination of three clauses; each clause in itself consists of a predicate - a statement that may or may not be true - relating an attribute of an object to a value. For example, in the second clause of rule 50's premise, MEMBF (meaning "a member of") is the predicate, CNTXT is the object - actually, this stands for "the current context, whatever it may be" - SITE is the attribute, and STERILSITES is the value with respect to which that object's attribute must satisfy the predicate. This clause would be true if the value of the SITE attribute of the current context is a member of STERILSITES (presumably a list of values).

The action part of Rule 50 consists simply of the identifier CONCLUDE followed by a statement of value of an object's attribute, possibly followed by a certainty index: here, the rule affirms that the IDENT attribute of the context is BACTEROIDES with a certainty of 0.7. Note that this fact could have been established by the user if he had answered positively question number 9, which asked "Enter the identity of organism-1". Mycin rules are triggered by values of attributes, and these values can be established either by user's answers or by rules' conclusions. Indeed, mycin's backwards chaining logic determines which questions to ask the user by determining which rules, if triggered, would restrict the potential conclusions the most.

Air Traffic Control Applications of AI Technology

This overview of the world of artificial intelligence has been, by necessity, very brief. It has not covered, for instance, any of the work done in a natural language processing, that is, the analysis of human language - written or oral - to extract specific information. We have not covered speech synthesis and recognition - a different problem than that of understanding natural language. We have not covered robotics, the discipline that deals with mechanical manipulators and touch sensors. Finally, we will only mention vision and image recognition, even though we believe there may be an opportunity for air traffic control (ATC) applications of artificial vision.

It seems that in order to do justice to the title of this presentation we should also briefly mention what we mean by ATC. By air traffic control we do not mean exclusively the activity of the man or woman behind the radar screen issuing vectors and clearances to aircraft and looking out for conflicts; we very specifically include all the activity that, combined, makes for a safe and efficient ATC system, such as planning the command and control structure of the system - that is, determining when and where information is transmitted, and when and where decisions are made - or selecting the set of airways that will constitute the preferential routes from two busy terminal areas in a particular complex weather situation. The possibilities for useful applications of AI technology to the world of ATC go well beyond the radar controller's screen.

Some of the technologies of AI can be of quite immediate applications; others may have to wait five, ten or even twenty years before they can be seriously considered. We will mention both short term and long term applications and will divide these immediate and future applications in a different way, namely two groups which we call visible and invisible.

Invisible applications are those where the AI component is hidden from the final user of the ATC product or system. Perhaps AI technology was used in the design, development or implementations of the system for economic reasons, or perhaps it is the only way in which to mechanize a certain function, but as far as the user is concerned, it is just another computer program.

In a visible application, on the other hand, the particular behavior of an AI product, as typified in the mycin example, is an essential part of the usefulness of the tool, and the user must be prepared and trained to use it in this way. In the invisible category we would like to mention symbolic programming, experimental simulation, radar tracking algorithms, and procedure generation. In the visible category we would like to propose a theoretical flow oriented command and control structure, an expert system to help select runway configurations, two very similar applications of visual scene recognition, and the controller's assistant concept.

"What?" you will say, "they are not going to talk about applications of voice recognition?" About the only application we can foresee for this technology is the simulation of pilots' voices - and ears - in a real time ATC simulation, and we are afraid that the available technology is not capable of doing even this. At the present time voice recognition and synthesis seems to be more of a solution looking for a problem, than a solution to an existing problem.

Artificial Intelligence and the Management of Complexity

The history of aeronautical technology has always been characterized by barriers, or measures of performance that were considered unattainable: transoceanic flight, stratospheric flight, blind flying, the sound barrier, the heat barrier, space. One by one these barriers have been conquered. We believe that the current barrier, the one performance limit we must conquer today, is the complexity barrier. Consider this: Charles Lindberg's aircraft, the Spirit of St. Louis, required 850 man-hours of engineering effort to design; the Lockheed C5A Galaxy transport jet took 49 million man-hours to design. As aircraft become more complex, and as the relationships between aerodynamics, propulsion, avionics, and even radar signature become more and more interrelated in determining the performance of the aircraft, this complexity, and the cost of designing it, will become greater and greater.

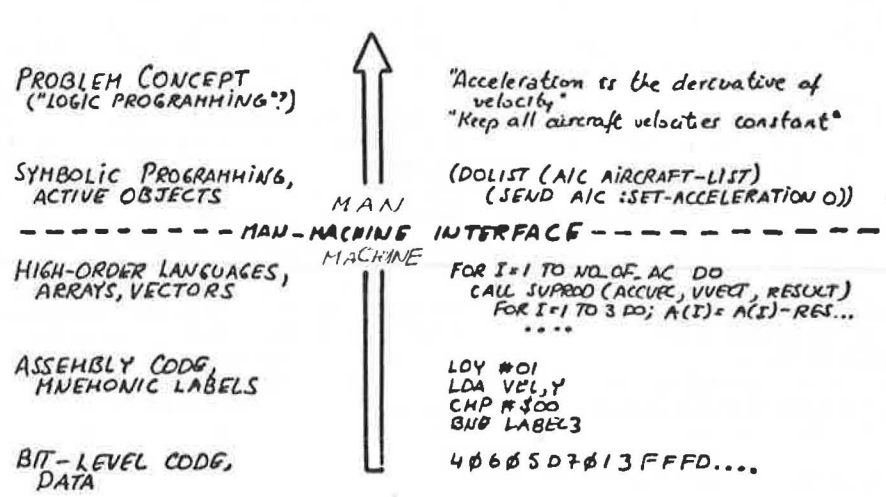
Nowhere is this more dramatic than in present and future ATC systems. the United States ATC system has already been dubbed "the most complex man-machine system in the world"; indeed, its complexity has reached a point where nobody quite knows how the entire system operates, and it is becoming more and more difficult to estimate what effect on the entire system the introduction of a new component, such as direct routings, will have.

Another area where the cost of this complexity is quite evident is computer software; it is a well established fact that the cost of developing a software system is not proportional to the size of the system: "two programmers can do in nine months what any of them could do in twelve months" is the popular proverb. A more detailed analysis of the additional costs incurred when a large software project is partitioned in N smaller components is N to the one and one-half power, and this, coupled with the decreasing cost of computer hardware has resulted in a reversal of the relative importance of hardware and software costs. Whereas fifteen years ago hardware costs for a large system were typically ten times larger than software costs, today it is software which is about ten times more expensive than hardware for a typical command and control system.

The differences in programming productivity are tremendous. While the industry standard for fully developed, tested and documented code ranges between 1200 and 2000 lines of code per man-year, project-wide averages of 20000 to 50000 lines are not uncommon in AI projects. In addition to the simple increase in single-programmer productivity, this difference is compounded by the reduction in the number of individual pieces in which a large project must be subdivided in order to meet the required schedule (the N to the one-and-a half power law), with overall differences in software cost of up to 100 to 1, for the same resulting software functionality.

The reason for this difference is actually quite simple. Programming is nothing more than the translation of the original functional specifications of the system to be designed into the simpler elements that can be executed in a computer. In the early days these were individual bits, representing either data or instructions, so that the entire translation process had to be performed by the human programmer. Next came the assembler or machine language which, while operating with the same machine-level elements, at least allowed the programmer to refer to them by names and symbols, rather than by anonymous numbers. The advent of the so-called high-order languages raised the interface to the level of vectors, arrays, strings and passive data structures, and produced what appeared to be a miraculous increase in programming productivity (Figure 10).

Figure 10. Man-machine interface in programming an ATC system.



High-order languages, even in their most complex form such as Ada, are still rooted in the Von Neumann concept of the computer as a sequential executor of instructions. Code and data, for example, are two distinct and unmixable elements, linkable only through the process of compilation. By comparison, symbolic computation removes itself one step further from the details of hardware, and allows truly abstract concepts to be represented and manipulated on a computer. Probably the most spectacular consequence of this increased level of abstraction is that the program itself, or code becomes simply one more abstraction, and thus can be directly manipulated by a program without the compilation or interpretation barrier of high-order languages.

And this is only the beginning. AI research is fast advancing in the direction of declarative programming languages, or rather, programming models, that allow the user to state the functional specifications for a computer system in extremely abstract terms without having to specify, for instance, the sequence in which operations have to be performed to arrive at the desired effect. These languages, while still many years away, may make Lisp look as mechanical and complex as high-order languages look in comparison to Lisp.

It is interesting to observe that while the attempts to build an automatic programming system during the early seventies were dismal failures, the same results are being arrived at by a diametrically opposite route. Instead of a very high level program that transforms any program specification to the detailed instructions that computer hardware requires, we are seeing computer hardware and software that operate at higher and higher levels of abstraction: a bottom-up approach, rather than the top-down approach of the automatic programming concept.

Of course, nothing comes free. This increase in the level of abstraction at which the machine interfaces with the human programmer entails an inevitable increase in the processing power required in hardware. But one should not look at this increase as inefficiency or overhead; in fact, this additional processing is performing an extremely useful function, namely the translation process from abstraction to machine bits and back, of both code and data. Therefore, we will have to learn to accept much higher computer processing requirements as a natural by-product of our increase in complexity. However, the continuing decline in the cost of processing, or, if you wish, the increasing performance of computer hardware will make it more palatable. The important point to consider is that the computer technology, both hardware and software, used today by AI researchers may become the only economical way of implementing very complex software systems in the near future.

Research Simulation Technology

Leaving behind the world of computer software, we find that some of the same problems that plague builders of large software systems also haunt designers of large human systems. Even if the ATC system used no computers at all, the flow of information, and the distribution of decision-making authority makes the system look very much like a gigantic computer, with procedures, rules, regulations, and letters of agreement being its program.

We have long passed the stage where the effects of major changes in procedures or technology can be evaluated effectively by simple analysis: simulation becomes the ultimate evaluation and verification tool. Unfortunately, building and running a sufficiently good simulation of a very complex system can be extremely costly.

Consider the difference between an aircraft simulator and, for example, the simulation of an advanced ATC controller station of the year 2000. While the basic principles of aerodynamics, structures, propulsion and so on cannot change radically from now to the year 2000, the same cannot be said, at least in principle, of air traffic procedures. There are few physical limitations to what can be displayed on a futuristic controller's screen. So whereas the aircraft simulation can count on a number of essential fixed elements no matter what the configuration of the experiment may be, the same cannot be said of an ATC systems simulation.

The traditional way of designing, implementing, and using large system simulators was this: a detailed specification was drawn of the fixed part of the system, that is, the part that is not expected to change from one experiment to another. Next, the user defined some bounds on the kind of experiments that would be run on the simulator. The simulator designer then would convert the fixed part of the specification to detailed formulations of the core of the simulator, which would include generation of large amounts of data that could be used to feed the expected experiments. Also, the behavior of the core system would be determined, as much as possible, by parameters that could be read from a data file in a simulation initialization time, so that the core could be tailored as much as possible to the particular experiment that was to be run.

The alternative to this traditional approach is to build not a core simulator, and an array of ad hoc extensions for each new experiment to be run, but rather a kit of building blocks with which a customized simulation can be built in a very short period of time. In other words, we not only accept, but actually encourage the notion that a new simulation will have to be built for each new experiment in ATC technology (Figure 11).

The key to this approach is the level of abstraction of these building blocks. Using symbolic programming techniques, it is possible to build blocks such as "VOR", "Aircraft", "Random Aircraft Generation Point", "Airport Runway", "Airway Intersection", "Radar", "Display Screen", and the like. Moreover, there can be many different types of these blocks, not only in terms of their performance parameters - you can do this in Fortran with initialization files - but even in the level of detail being simulated (Figure 12).

For example, the MIT Flight Transportation Laboratory is currently designing a building block kit which will allow the experimenter to intermix three very different levels of simulation at the same time: a Level I, where the smallest geographic unit represented is a control area, say several sectors large, and aircraft dynamics consist only in movements from an area to an adjacent area (Figure 13). At this level of detail, the entire continental United States could be modeled, with some 2000 aircraft, with very little effort required to set up the experiment. A Level II would look into the actual geometry of the airway structure, as well as direct routings, and be able to model individual sectors. At this level of detail, the position of each individual aircraft along an airway or along its direct route would be modeled, but not, for example, the effects of individual radar vectoring. The maximum number of sectors that one would like to model this way is probably ten or fifteen, with a total of one to two hundred aircraft, enough to analyze problems relating to the communications and handoffs between two centers. Finally, Level III of simulation detail would look at individual aircraft

Figure 11. ATC customized simulation building blocks.

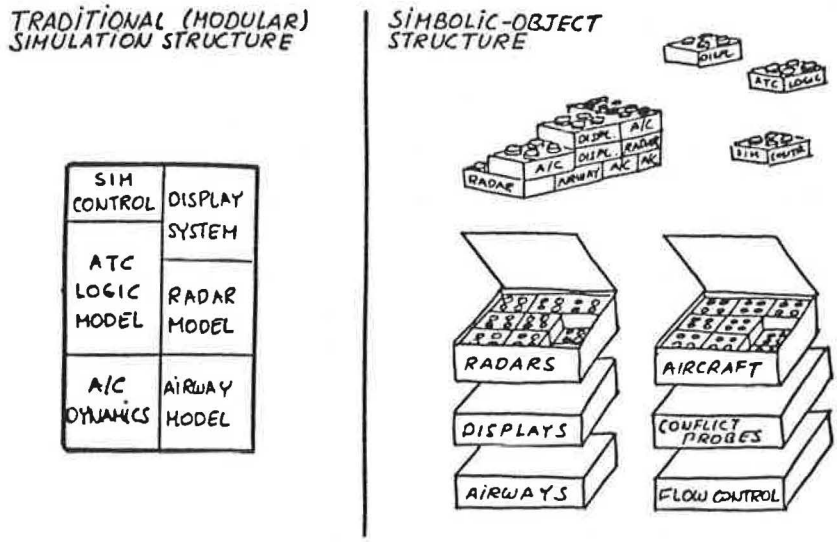


Figure 12. Typical symbolic-object simulation.

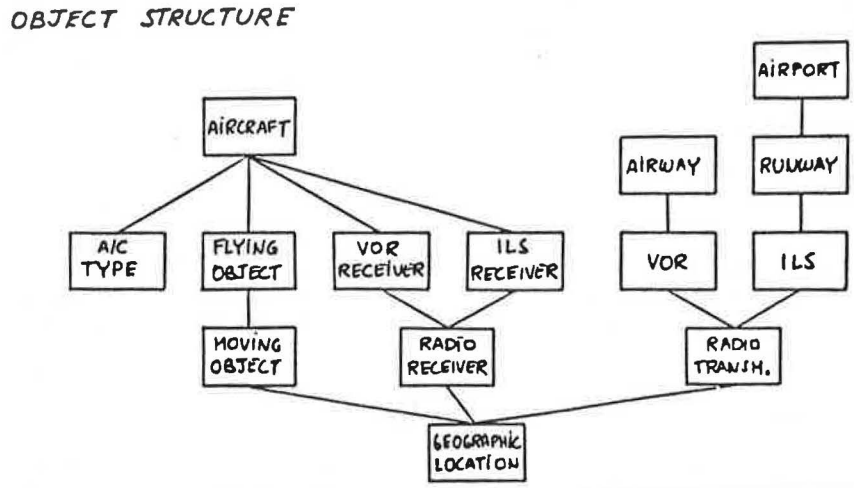
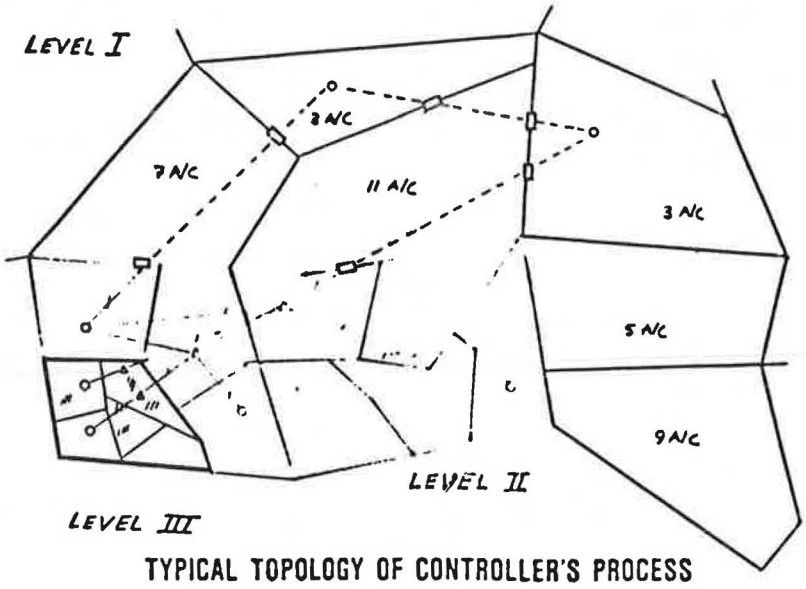


Figure 13. Example of levels of simulation at the same time.



dynamics and the performance of radar sensors, and would be the level of detail at which to look at problems such as simultanous instrument arrivals to closely-spaced parallel runways, or the sector-to-sector interactions for a maximum of, say, three sectors and thirty or forty aircraft.

This building block kit would then include not only three levels of airspace models and three levels of aircraft models, but also different display format for each level. The important feature of this approach is the possibility, if designed correctly, to run a simulation where the entire country is modeled with Level 1 elements, except for two centers, which are modeled with Level II elements, and have within these two centers two or three sectors modeled with Level III elements.

Object-oriented and symbolic technology are capable of solving the problem of interfacing these rather dissimilar objects together. Consider a flow control algorithm that wants to know how many aircraft are in a certain area, the smallest Level I unit of airspace. In traditional programming, the programmer would have to know the location of that number in whatever data structure contains that information for a Level I area, but would probably have to write a subroutine to obtain that information from a Level II center, since it would have to add all the aircraft in each of that center's sectors. With object-oriented programming, the burden of providing any information about an object is shifted from the seeker of the information to the supplier of the information.

The technique in question is called message passing; each object in the kit is known to respond to a certain number of requests, or messages. These requests can either ask for information about the object, or ask that the object perform some action that has a side effect, such as displaying a symbol on a screen. All the interactions between objects must be through these publicly advertised messages. Part of the effort required in designing such a simulation is to define what kinds of messages each object should be required to handle.

Once this is decided, though, the task of inter-object communication is enormously reduced; if both Level I areas and Level II centers are required to reply to the message "how many aircraft do you have now", it does not matter to the object requesting the data whether this data is obtained by simply looking it up somewhere, or by laborious computation: it simply is returned in response to the message. If the internal makeup of an object must be modified - say, in response to the requirements of a new experiment - only its way of handling its incoming messages must be modified, whereas in the traditional technology every object that could possibly interact with the modified object would have to be modified as a consequence of this change.

The development of this simulation architecture is the most exciting ATC-related project at the MIT Flight Transportation Laboratory in the last decade. If successful - and there are a number of major technological obstacles still to overcome - it may enable for the first time the testing and evaluation of truly advanced ATC concepts in a sufficiently realistic environment, at reasonable cost.

The concept of building a real-time ATC simulation based on software building blocks as just described has been demonstrated at the Flight Transportation Laboratory, where a full scale Level III simulator using this technique is in daily use. The largest technology risk associated with this simulation is related to its hardware; in addition to the building block software approach described, it is designed around a building block hardware architecture; the same message-based interaction technique that allows different kinds of objects to interface in a homogeneous manner will also allow these objects, and the functionality they carry, to reside in different processors, with some limitations, so that the exact number of processors available to run the simulation is invisible to the software, although, of course, the resulting performance will be very visible to the user.

This will also allow incremental growth in the capabilities of the simulator, as more processors and display hardware are added without the need for software recoding, but is dependent on very recent, and still untried advances in symbolic computation hardware.

An Expert System for Runway Configuration Management

Curiously, there are fewer opportunities for classical expert systems such as mycin in ATC than one might expect. Indeed, there are few circumstances where accumulated knowledge, as opposed to skill or ability, determines the performance of a control function.

Perhaps one of the most promising short-term applications of classical expert systems may be to the problem of runway configuration management, that is, the selection of what runway configuration to use under changing weather and flow conditions. Complex airports, such as Chicago, or the New York City Metroplex, have hundreds of possible runway and approach configurations. The problem consists in selecting which configuration to use, and, more particularly, selecting when to perform a configuration change. The relative timing of the arrival of a front at the airport terminal area with respect to the peak traffic hour may make a difference as to whether the runway configuration change should be performed in advance, or delayed with respect to the weather-optimum time. Moreover, weather at other airports may affect the normal traffic pattern at an airport so that, for example, a snow storm approaching the Boston area from a westerly direction requires a different runway configuration change strategy than one approaching from the northwest, since the former will hit New York before Boston, therefore causing potential diversion of traffic from the New York City area.

This simultaneous consideration of multiple contradicting factors, some of which may be the result of many years of experience and observation at the station in question, lends itself ideally to mechanization as an expert system. Indeed, the MIT Flight Transportation Laboratory is developing such an expert system, under the code name Tower Chief. This name was selected to bring to mind the notion that the Tower Chief is usually the senior - and therefore the most experienced - controller in that facility, and therefore would be the ideal person to make runway configuration decisions at all times, not just when he is the actual shift supervisor. By capturing his expertise, the expert system would make available to any supervisor having to make such decisions the expertise and accumulated knowledge of the senior person.

Actually, such an expert system would be capable of storing knowledge and associations furnished by a number of individuals, and therefore be of use to the Tower Chief himself, specially in its ability to be comprehensive in examining all the knowledge elements pertinent to the current state of affairs. On the other hand, we dislike the name Tower Chief since, in addition to the concept of

wisdom and experience, it also calls to mind the concept of authority, or responsibility. There is therefore the danger of concluding that such an expert system, by virtue of its superior data base, is able to make superior decisions than a human in this situation. This is clearly not so. In fact, beyond the assurance that the expert system has systematically tested all the knowledge contained in the data base, the greatest benefit that the shift supervisor can derive from the use of Tower Chief is not the final conclusion or recommendation that it may make regarding the runway configuration changes to select, but rather its compability to display the logical process that leads to that conclusion. This display can be used not only to help make a final decision, but also to enrich both the expert system's and the human's knowledge base; therefore, we would have preferred to title this project supervisor's consultant, but it is a little late for this, so we will continue to call it Tower Chief.

Some technical problems must be resolved before rules and knowledge can begin to enter a Tower Chief prototype system. As with all knowledge based systems, expert or not, the work begins with the construction of logic abstractions capable of representing, both to a computer and its user, the elements of knowledge in the particular field. For Tower Chief these may be runway, prevalent winds, primary flow direction, etc. with again, both data and functionality being associated with these abstractions. This is the knowledge engineering phase, and is now under active development for Tower Chief at the Flight Transportation Laboratory.

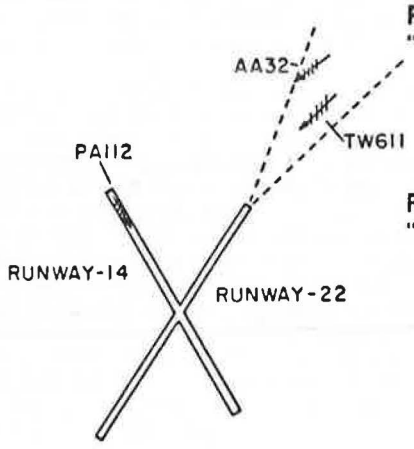
Simultaneously with the knowledge engineering phase, an expert system systems design must be carried out. This is the design of the process by means of which the abstractions will be entered, searched, activated, processed, and displayed in the operation of the expert system. There are a number of classical methodologies, such as forward chaining, where as many of the rules as may possibly be achieved given the established facts are invoked, until all the rules have been used, and mycin's backwards chaining, where a number of hypotheses are postulated and tested by means of the rules, until as many of them as possible have been weeded out. Other classical techniques address the method of incorporating rules into the knowledge base, requesting specific data items as the hypothesis tree is traversed to reduce the number of branches that must be explored. The collection of techniques and the software used to implement them are referred to as expert system cores.

A small but growing industry of pre-fabricated expert system cores offers a large number of more or less off the shelf software systems. These cores consist of a general-purpose structure for representing knowledge, and the inference engine or logic that drives activation of the rules to achieve the final objective. Along with these features, some of these systems also come equipped with fabulous claims about the speed and ease with which useful expert systems can be built around them.

Unfortunately, these claims are usually exaggerated for two reasons: first, because experience has shown that rule-processing procedures are much less universal than previously thought; second, because even if an existing core is adequate to perform the rule processing required in a particular problem, a significant knowledge engineering effort is usually required to case the particular knowledge relevant to the problem in the forms required by the expert system core.

Tower Chief is the second ATC-oriented prototype expert system developed at the Flight Transportation Laboratory. The first, known simply as Rule System One, or RS-1, was only an experimental system in which conventional algorithms could be reimplemented as rules, and was developed to gain familiarity with expert system techniques, and not to demonstrate any useful function (Figure 14). RS-1 showed us, for example, that ATC problems are particularly ill-suited for prefabricated expert system cores. In RS-1, data, or rather assertions about the objects known to the system, arrived in time-sequenced frames, corresponding to entire revolutions of a terminal radar antenna; thus the assertion base, the data base of statements asserted to be true about the objects, was continually evolved. Moreover, rules may refer not only to current assertions, but also to past assertions, or even changes in assertions, as for instance: "If aircraft-1 appears to be on a base leg, and it was previously affirmed to be on final, something is wrong". Among the interesting consequences of the RS-1 work we found that the concept of past, as applied to

Figure 14. ATC experimental expert system RS-1.



RULE-1 (AIRCRAFT1, RUNWAY1):
"IF AIRCRAFT1 IS-TRYING-TO-LAND-AT RUNWAY1
IS TRUE AND
AIRCRAFT1 IS-CLEARED-FOR-APPROACH-TO
RUNWAY1 IS FALSE THEN
ALERT USER"

RULE-2 (AIRCRAFT1, RUNWAY1):
"IF AIRCRAFT1 IS-CLOSE-TO RUNWAY1 AND
ABS (AIRCRAFT1 COURSE - RUNWAY1 HEADING)
LESS THAN ϵ AND
ABS ((RELATIVE-BEARING (AIRCRAFT1 POSITION)
(RUNWAY1 TDZ)) -
RUNWAY1 HEADING) LESS THAN ϵ AND
AIRCRAFT1 ALTITUDE-AGL LESS THAN 1500 THEN
ASSERT AIRCRAFT1 IS-TRYING-TO-LAND-AT
RUNAWAY1 TRUE"

RULE-3 (AIRCRAFT1): ASSERT AIRCRAFT1 COURSE $\tan^{-1} \frac{(x1 + x2 + x3)}{(y1 + y2 + y3)}$

computer implementations of knowledge, is more complex than previously thought.

Symbolic computation has taught us that the concept of equality is more complex than the simple equality of numerical values of Fortran. For example, a simple chair and an armchair are clearly not equal, while two identical armchairs are, to a certain degree, equal, although they are two different chairs, two different actions of equality. Similarly, we have two different notions of past. Suppose, for example, a rule which estimates the general direction of an aircraft track; this rule may ask the assertion base for the previous heading of the aircraft in order to compare it with the current heading. But suppose that, during the previous four-second revolution of the antenna, insufficient valid transponder hits were received and a missed reply was declared for that target during that antenna revolution; what should be answered to the question "what is the previous target data?" One possibility is to answer "not known", since there was no reply on that antenna pass. But another is to return the target data for the last antenna pass during which there was valid data. In a way, both are previous data, but the answers may be quite different.

The consequence is, of course, that there are at least two different pasts, one relating to the sequence of known data, independently of the time at which it was asserted, and another relating to a sequence of instants of time. Such a feature was not available in off the shelf cores at the time the RS-1 effort was started.

In addition to this passage-of-time problem, Tower Chief will also be subject to three more time related problems. First, the elements of knowledge that Tower Chief will handle will have themselves a time component, similar, but more complex, than the time related questions asked by mycin.

Second, the goal of this expert system is really a program, or timed sequence of runway configuration changes, so time is one of the components of the answer, as well as of the data used to arrive at the answer; nobody has had any significant experience in designing expert systems that deal with time as one of the parameters of the goal.

Third, and this is a problem faced by all expert systems whose answer is required in real time - the search for answers may be terminated by the time available, rather than by exhaustion of the search, as in mycin, where the time required to arrive at the answer is not really important, as long as it is reasonable. There is little experience about time constrained expert system performance. Indeed, expert systems share with some operation research methods the property that, while monotonic, the rate of improvement of the answer may vary widely with time. In some cases an excellent answer may be arrived at very quickly, with only marginal improvements afterwards. In other cases, the bulk of the solution improvement may only be achieved at the very end of the search, so that an early termination may produce a very unsatisfactory answer. It is not known at this time if the amount of processing required by Tower Chief will be such that time-terminated processing will be required; if it is, its performance may depend on new developments in solution search techniques which guarantee uniform solution improvement with time. As an aside, one of the methods that have been proposed to achieve this uniformity involves the intentional randomization of the search procedure, in a Monte Carlo like process.

Two Simple Applications of Mechanical Vision In ATC

An entire field of research in artificial intelligence is that of visual scene recognition, that is, the processing of raw data from, say, a television camera or other means of converting visual information into bits, with the purpose of identifying objects, positions, three-dimensional shape, and even higher order relationships, such as attachment between objects or their constituent materials.

At first glance there would seem to be no obvious application of this robot vision capability in air traffic control, unless one wished to build a robot tower controller or a robot pilot. Actually there are two very good possibilities, one on the ground, and one in the air.

A useful ground system based on mechanical vision and scene recognition would be a low cost, totally passive LIDAR, or Light-based Radar. Such a system would consist of two, perhaps three television cameras mounted on fast remote-controlled tilting and panning heads, and equipped with fast zoom lenses. Controlled by a computer with visual recognition software, this system could act as a VFR radar in congested small general aviation airports whose traffic density changes from being higher than that of Heathrow during fine VFR conditions, to practically nothing as the weather becomes IFR. Visually scanning for aircraft, this system could present to the local tower controller a plan view display of the aircraft within the airport's traffic area.

In its simplest form this system would periodically scan the horizon surrounding the airport and create a visual map of the fixed features around the cameras: trees, buildings, hills. Some of these features may change periodically, such as the foliage of the trees, but just as in a modern radar's clutter map, they can be immediately recognized by their very, very slow rate of change.

Real scene recognition begins with slow, but really dynamic objects, such as clouds and birds. Clouds have such a characteristic texture, size, and speed that it should be trivial to separate them from aircraft targets. How can this system distinguish a bird at five hundred meters from a light airplane at five kilometers? One possibility is radial velocity: the bird at five hundred meters can move faster across the camera's field of view than a similar-sized aircraft target.

In addition to acquiring all this information the system has some unusual potential for presenting the information to the controller. For instance, instead of the usual bars we are accustomed to in high-intensity radar displays, we could have a small picture of the actual aircraft, in color, obtained by the system's cameras, and processed by the computer so that at any time in that aircraft's flight that picture should look just like what the controller should see with his binoculars were he to look for that aircraft.

Now we have a system that not only is more sensitive than a human controller in detecting and processing visual targets, but may even provide him with additional information about the target that a conventional radar certainly could not. And being only software, it is a cheap system to produce in large numbers, so as to offset its probably large software development cost.

An Abstract Concept of Flow Control

The next concept in air traffic control that we will consider is not a gadget like Tower Chief or the visual radar, but actually a concept. It is related to artificial intelligence because it is the result of building abstract representations of knowledge, capable of being implemented on a computer, but also independent of any computer implementation. Indeed, they could very well be implemented as procedures, with humans performing all the information handling and decision making.

These abstractions are models of how a flow of aircraft could be regulated by control elements that interact only with their neighbors; at what level this flow control would be carried out is immaterial. The test prototype we have implemented in our computer at the MIT Flight Transportation Laboratory operates at the tactical, terminal area level; but the concept could equally well be implemented at the central flow control level. It is far too early to decide whether these abstractions would be of any use in a future ATC environment or not. Our purpose in presenting this work is to show a different kind of product of artificial intelligence thinking in air traffic control research.

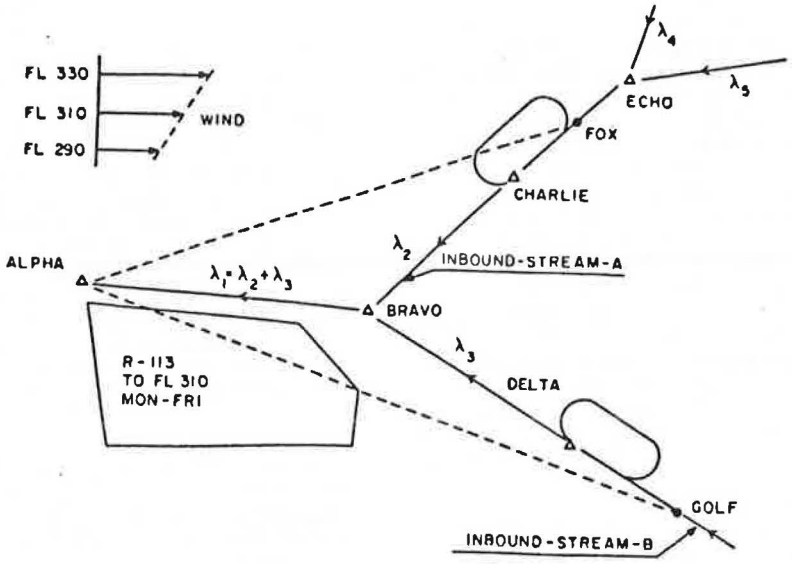
The development of these abstractions began as an attempt to state, in knowledge representation terms, the classical time based metered merge control problem, which can be simply stated as follows: merge two streams of incoming aircraft with random interarrival times to form a single output stream with uniform aircraft separation (Figure 15). This is usually performed by assuming an ideal conveyor belt of time slots, and by assigning aircraft from both incoming streams to a slot in the conveyor belt, and then maneuvering the aircraft - in the time dimension, hence the name time-based merge - to their assigned slot. This maneuvering in time may, of course, require complex maneuvering in two-dimensional space. (Figure 16).

The picture is a little more complicated when not two, but a number of incoming streams must merge into a single one. Each route begins at one of the sources; the routes merge in pairs, until a single path arrives at the sink, thus creating a binary converging tree.

Flow control is only one of the tasks to be performed by the ATC system. Indeed, separation assurance is by far more important, in the short term, than orderly flow of traffic. For a number of technical, operational and historical reasons responsibility for separation assurance requires that ATC functions be divided into small sectors under the authority of a single human controller, as opposed to a central control authority. This federated approach, which is optimal for separation assurance and responsibility accounting, conflicts with the centralized approach of traditional flow control algorithms. In a federated approach, each control element, that is, each controller, interacts mainly with his immediate neighbors, rather than with a centralized arbitrator. Handoffs are initiated, accepted, or rejected on a one-to-one basis, and not as a result of the decision-making of a central authority.

For this reason, flow control procedures are difficult to implement and interface with in a federated ATC environment. It would be interesting to develop and test a flow control approach that operated as a number of independent elements which interact only among neighbors, in the same way tactical ATC elements do. This approach, developed at MIT's Flight Transportation Laboratory, is called the Metered Merge Control Element, or MMCE, concept. Again, it is too early to decide if this approach has any merit, and is presented here only to illustrate the kind of product that can be developed using the AI approach to computers.

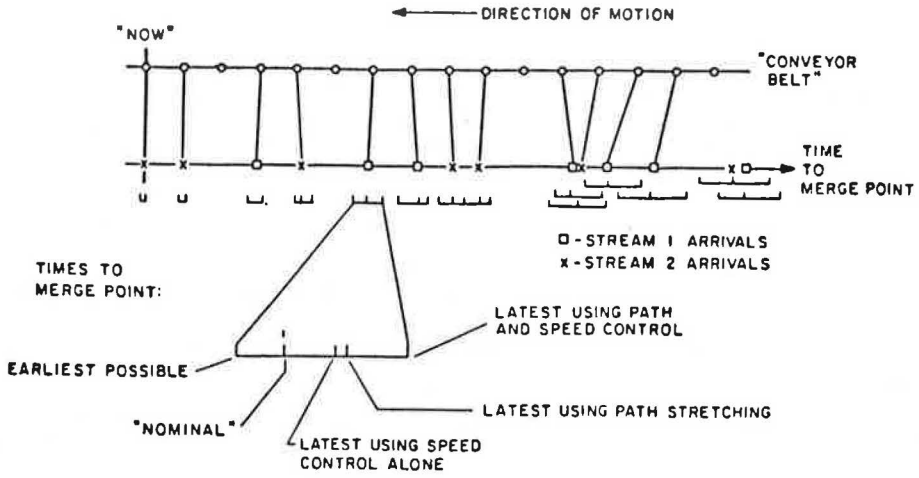
Figure 15. Stream merge problem of an ATC system.



Conceptually the MMCE consists of the following elements: two entry gates, a single exit gate, and two nominal transit times from each of the entry gates to the exit gate. While it is useful to visualize the MMCE as a Y-shaped merging path, the geometry of the MMCE is irrelevant to the concept, except insomuch as the transit times are related to the size and shape of the paths (Figure 17).

Connected to the exit gate, each MMCE has a downstream correspondent which can be either another MMCE or, in the case of the last MMCE of the tree,

Figure 16. Time-based metered merge-base model of an ATC system.

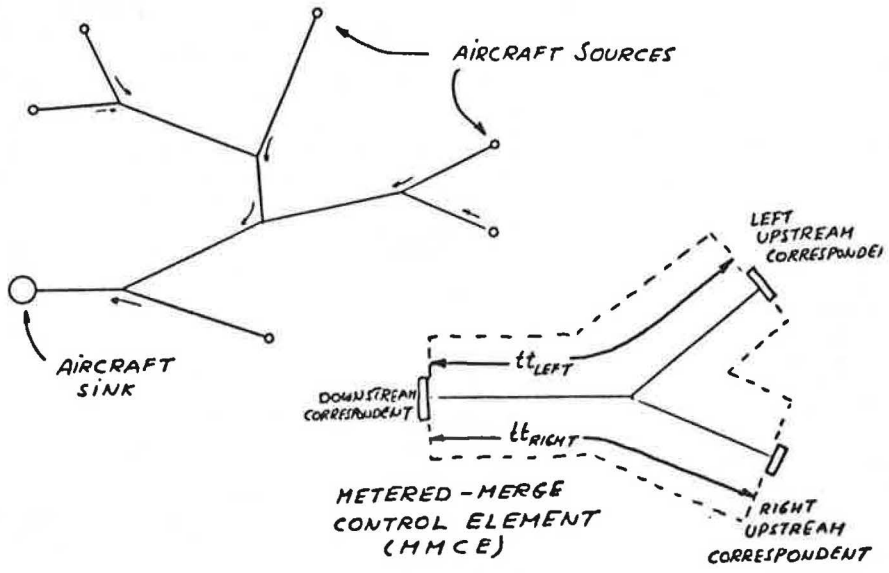


the aircraft sink. Connected to each entry gate is an upstream correspondent, either another MMCE or, in the case of the first MMCE in the tree, the aircraft sources. Sources, MMCEs, and the sink comprise the entire metered merge flow control abstraction. This abstraction is independent of the scale of the problem: it could be the terminal area around an airport, with the sources being the feeder fixes, and the arrival runway; or it could be an enroute problem, with the sources being originating airports and the sink the destination airport's terminal area. In any case the operation of the abstraction is as follows.

When an aircraft appears at a source, its existence is immediately made known to the MMCE immediately downstream of this source. In the absence of any flow control, that aircraft would reach the MMCE's exit gate at a time which is equal to the time at which the aircraft appeared, plus the nominal transit time through the MMCE's right or left branch, as appropriate. Therefore, that aircraft should appear at the entry gate of the current MMCE's downstream correspondent at that time. This information is passed on by the current MMCE to that downstream correspondent, who then performs the equivalent computation and passing of the information to its downstream correspondent. Finally, the ultimate downstream correspondent, the sink, is told that an aircraft would nominally reach it at a time equal to the current time plus the sum of the nominal times through all the appropriate branches of all the intervening MMCEs.

At this point the sink has to perform its own decision-making, which may include previously received notifications of incoming aircraft. The result of this decision-making is a desired arrival time for that aircraft, which may or may not be the nominal arrival time. This information must then be made known to all the MMCEs that the aircraft must traverse to get there. Since the sink only has communications with the last MMCE, this element receives the desired arrival time at the sink for that aircraft.

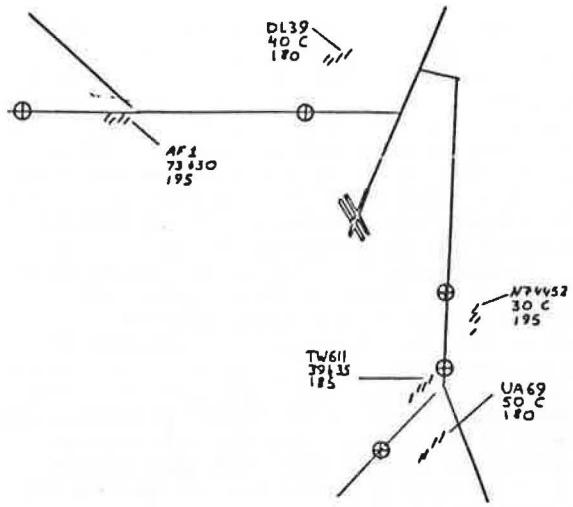
Figure 17. MMCE flow control procedure.



The process used to propagate the nominal arrival time downstream is reversed, in that the MMCE's nominal transit times are subtracted from the desired arrival times before submission to the next upstream correspondent. Finally, the first MMCE (the one currently responsible for that aircraft) receives the time at which the aircraft should leave its exit gate so that, flying at the nominal speed through the remaining MMCEs, it would arrive at the sink at the time that the sink desires it.

Actually, this upstream propagation of information is not as symmetric with the downstream propagation as we described it. Indeed, when propagating the information upstream, each MMCE has to send it to its right or left upstream correspondent, as appropriate, a decision-making not required when propagating the information downstream.

Figure 18. A radar controller's display of the MMCE concept.



In the Flight Transportation Laboratory implementation, the MMCE concept is used to drive a Radar Controller's display. In this display, the MMCEs are made to correspond to actual converging ATC paths. In this way each controller is given an indication as to how early or late the aircraft is with respect to the ultimate sink's wishes (Figure 18). This display concept, or conveyor belt had been proposed before, although it has never been mechanized, even experimentally, beyond the final approach path. It is clear that this kind of display could be constructed without the need for MMCEs, downstream ripples, upstream ripples, and the like.

While the development of this abstraction does not imply its computer mechanization - it could be mechanized, for example, as a series of controller-to-controller interactions - we are able to simulate them, and therefore perform experiments with them, using software objects in Lisp in the MIT Flight Transportation Laboratory's symbolic ATC simulator. A number of instances of sources, sinks, and MMCEs can be created, linked, and positioned interactively. Image objects corresponding to the MMCEs' nominal paths and the previously described slots are created and manipulated as easily as numbers of a calculator or characters on a word processing system.

A Distant Dream: The Controller's Assistant

Finally, and as an example of a truly long-term possible application of AI technology to air traffic control, we would like to propose the idea of a personalized controller's assistant. This device would consist of a knowledge base made up of four parts: a general part reflecting the generic kind of controller know-how that would be reflected, for example, in the Controller's Handbook, or in training material; a second part, at a higher priority level than the first, would include position-dependent knowledge, such as the route and airway structure pertinent to that facility, letters of agreement between facilities, and the like; the third part would include the daily weather, notam and similar information, while the last part would be made up of the individual controller's performances and personal techniques.

Exactly what functions such a system could perform is not very clear at this time; one possibility is to act as a dummy of the controller, that is, display for his benefit what control actions the clone would take. By periodically observing that dummy controller the human controller could detect his own blunders, especially missed control actions, early enough to take effective corrective action.

If such a feature is to be a real help, rather than an additional burden, it is likely that the display of such dummy directives would have to be at a rather high level of abstraction. For example, rather than the clone displaying the command "TW611 turn right heading 220", to which the human controller may think "Why is he doing that?", the display should read something like "I would like to send TW611 west to make him a little late on his turn to final, or else he is going to be too close to that heavy ahead of him".

The key characteristic of such a system would be its personalization capabilities: personalization with respect to the position being assisted, the current weather, navaid and traffic information, and most important, the individual controller. The controller's individual knowledge base could, persumably, be part of his personal equipment for the duration of his career. If we may be allowed to dream for a moment, we can imagine the days when the controller, upon taking over a position from the previous person, would insert his or her magnetically-coded ID card on the console, to indicate to the system that his personal knowledge base is to be used. This knowledge base would replace the previous controller's personal set of rules, and interact with the facility's rule set, as well as the knowledge of the day which was entered by the same shift supervisor that briefed the incoming controller on the day's situation. Thus, there is a one-to-one correspondence between one element of the knowledge base and the controller's basic training, knowledge of the local environment, personal controlling style, and knowledge of the current traffic, weather and facilities situation.

What the form of this knowledge would have will have to wait for the appropriate knowledge engineering to be performed. We can only venture to suggest that it will involve abstract concepts both intuitively obvious to the human and manipulable by the computer, similar to the geographic location and intersection objects of our symbolic ATC simulator. The collection of abstractions, which would include both objects and actions, would in effect create a rich, unambiguous and intuitively attractive language which could be useful not only for humans

and machines to communicate, but even for human-to-human communications, much in the same way that the language Lisp is today used not only to program, but also to describe logical process in scientific publications.

The same uncertainty about how knowledge would be represented in such a system also applies to what kind of inference engine or rule-processing logic it should have. To begin with, several simultaneous goals may be required, and these goals may be more complex than the simple diagnosis-seeking of the mycin or the runway configuration change program of Tower Chief. Certainly today's expert system technology is not sufficient to achieve this functionality.

A Final Caveat

As ambiguous as all these promises are, they appear to hold a lot of promise for performance that we know cannot be achieved by today's computational techniques. It is also fair, however, to point out some potential problems, principally that of software verification and validation. A significant part of the cost of today's software is associated with achieving a satisfactory degree of confidence that the behavior of the software in a system as critical as the air traffic control system will be correct. The cost of this validation increases, of course as the complexity of the desired behavior increases; the problem with the personalized algorithm just described is not only that its behavior is radically more complex than that of any software ever used in ATC automation, but that its behavior cannot, by definition, be completely known and specified *a priori*.

This problem is not unique to the controller clone idea. Indeed, imprecise *a priori* knowledge of the behavior of the system seems to be a fundamental feature of most AI-oriented devices. What is the solution, then? Abandon this class of software as untestable? Abandon the notion that we can validate the software to be used in air traffic control? Both extremes seem unjustified. A new concept of software reliability must be developed, a concept more sophisticated than just the idea that it meets the prescribed specifications. For example, the notion of a software defect could be organized in various categories. Category one would be a software defect that simply and catastrophically causes the entire system to stop functioning. Probably we can devise methods for testing against that type of bug, no matter how complex the software and the expert system rules become.

A second category of bug would involve a less than perfect solution to a problem, such as not finding a solution to a specific problem. In this case it is clear to the user that the system is not functioning properly in that particular instance, but in all likelihood it will function properly on the next problem. This we would categorize more as a performance limitation of the technology than a real bug, and the difficulty here is that we cannot predict, therefore specify, what the performance of an AI-based product will or should be. We will have to learn to live with this type of software deficiencies.

A final and perhaps the most devastating type of bug would be one which involves a definite malfunction whose effects, however, are not immediately apparent to the user. Such a defect, for instance, would involve making decisions about an aircraft on final approach using data pertaining to another aircraft on final approach. Since the aircraft are in similar situations, the control actions suggested may look reasonable for the aircraft in question, even though they were based on information about the wrong aircraft.

How would one be protected from such defects? Perhaps a way out would be to implement software redundancy in the same way as today we implement hardware redundancy to protect against hardware malfunctions. The notion of redundant software is, however, very different from that of hardware redundancy. While two identical ILS receivers do offer a significant amount of protection against receiver failure, two copies of the same program offer no protection against a programming bug. Indeed, programs, or, in the case of AI products, the rules or other language data that determine the behavior of the program, must be independently developed, implemented and tested, to offer any degree of protection.

We are at the very infancy of software redundancy. With today's programming technology, exhaustive validation and verification are cheaper than redundant software development. With the next generation software technology and systems complexity it is possible that redundant software development may be the cheapest way, or may be the only way, of gaining confidence in critical software.

To summarize, artificial intelligence is a source of extremely powerful tools and ideas, and in particular, it opens up a new viewpoint on the use of computers for any kind of applications. One should not expect miracles from this technology in the near future, except perhaps in the areas of software productivity and simulation technology. We would like to compare the state of AI today with that of the transistor in the late 1950s. At that time there was little a transistor could do that could not be done with vacuum tubes. Admittedly, the transistor was a little smaller and used a little less power than a vacuum tube, but in many respects, such as frequency response, it was in fact inferior. Yet today it would be a little hard for us to walk around with a wrist watch that computes inverse trigonometric functions if it were built with vacuum tubes, even if we had a long enough extension cord. So, sometime between 1960 and 1985 the mere quantitative advantage that the transistor had over the vacuum tube was transformed into an insurperable qualitative advantage. Perhaps we will wake up some day in the year 2000 and realize that sometime between 1985 and 2000 the mere quantitative differences between artificial intelligence and conventional use of computers was also transformed.

Acknowledgments

Research described in this paper has been sponsored by the Federal Aviation Administration and the Transportation Systems Center of the U.S. Department of Transportation. The authors wish to recognize the important contributions of the following individuals: Professor Robert Simpson, Mr. Lyman Hazelton and Mr. Jim Butler of the MIT Flight Transportation Laboratory; Mr. John Fabry of the FAA Technical Center, Atlantic City; Mr. Richard Wright of the MIT Transportation Systems Center, Cambridge, Massachusetts; and Messrs. Paul Neumann and Steve Alvania of the FAA Headquarters, Washington, D.C.

DEVELOPMENT ENVIRONMENT FOR AN ATC EXPERT SYSTEM

David A. Spencer
Massachusetts Institute of Technology
Lincoln Laboratory

This presentation concerns a development environment for an expert system to control air traffic. The expert system itself will not be discussed, as work on that system is just starting. This effort is sponsored by the Federal Aviation Administration (FAA) under a contract that started in 1983 to look into artificial intelligence (AI) applications in air traffic control (ATC). Two projects have been emphasized, an air traffic control project and a weather radar data interpretation project. This presentation will discuss the ATC project, and the talk by Steven Campbell will discuss the weather project.

Overview

The presentation by Paul Neumann of the FAA described the need for increased automation in ATC. This presentation will briefly address that issue and then discuss AI techniques, particularly expert system techniques, and why they might be applicable to ATC automation. The presentation will then examine the system which the MIT Lincoln Laboratory has created to support the development of an ATC expert system. Its function is to provide an environment in which an air traffic expert system can be tested against traffic scenarios. Finally, several snapshots of one of these traffic scenarios will be shown. This will demonstrate the development system's current capabilities and also give an indication of the traffic control situations which the expert system must eventually handle.

The Need for ATC Automation

The motivation for additional automation is that the amount of traffic is expected to increase significantly, perhaps by as much as a factor of 2, over the next 20 years. It is too expensive to increase the number of controllers proportionally. Furthermore, adding controllers implies reducing the size of the sector controlled by each controller. At some point the increased intersector coordination workload takes away much of the benefit of the reduced traffic load per controller. It is hoped, therefore, that additional automation will increase the individual controller's productivity in the sense that he will be able to safely handle larger traffic loads without any increase in perceived workload or stress.

The FAA has been implementing automated controller aids for precisely this purpose at least since the late 1960s or early 1970s. The current automated systems in fact allow controllers to handle much more traffic than they could have under earlier non-automated systems. However, the current ATC computers perform mainly clerical functions. There are some exceptions, but by and large they perform clerical operations that assist the controller by performing calculations and providing information to him. The motivation for looking at AI is the belief that there is going to be more automated decision making in ATC.

Artificial Intelligence Technology

AI technology can usefully be looked at from at least three viewpoints. One viewpoint is system or application oriented. From this viewpoint AI technology can be divided into the areas of knowledge-based or expert systems, speech and natural language, vision and image understanding, and robotics (by which is meant the mechanical aspects of AI not included in the other areas). Another viewpoint emphasizes the basic underlying techniques on which all AI systems are based. The major categories here are representation of knowledge, reasoning (which includes search, planning and problem solving, as they are all closely related), pattern recognition (although numerical pattern recognition techniques are usually deemed not to be AI), and learning.

The emphasis of the MIT Lincoln Laboratory air traffic control work is on knowledge-based expert systems. There is some interest in speech and natural language as ways of providing an interface to such an expert system, but the Lincoln Laboratory is currently not doing any research in that area. The techniques of interest are representation of knowledge, reasoning/planning/problem solving, and to some extent pattern recognition. No work involving learning techniques is currently planned.

A third viewpoint on AI technology emphasizes the software development methods used by AI researchers. These include both specialized hardware, such as Lisp machines, and powerful software tools. These represent important developments for software engineering as a whole, not just for AI.

Expert Systems

Expert systems have an expert level of problem-solving ability within a narrow domain. They incorporate knowledge of human experts in a form that, ideally, is uniform and easily modifiable. The reason for wanting these characteristics is that the expert system development process involves presenting problems to the system, having a human expert criticize the system's solutions to those problems and then quickly localizing and modifying the items of information that caused an erroneous conclusion to be drawn. This can be contrasted with a more typical software development process where the program is represented in a flowchart-like fashion, and where a particular piece of knowledge about the world may be represented diffusely throughout the flowchart. In an expert system this piece of information is represented as one rule or one fact, one piece of knowledge. If you modify that one piece of knowledge you modify its use throughout the system.

The expert system development process, therefore, is a form of rapid prototyping. You can quickly develop a program that produces results. The expert can criticize these results, and in a matter of a few years to a few days it is possible to try several problems, criticize the answers, make necessary modifications and quickly converge to a system that performs properly. Contrast this with the typical development process for any large military or air traffic software system. This is a promising approach for the initial, more experimental stage in the development of any large software system. While such a prototype may not be operationally qualified, due to slow response times, or excessive resource usage, or lack of some functions, it can be reimplemented knowing that the functional requirements and algorithms are well understood.

Some expert systems have the ability to explain their reasoning to the human expert in a format that is easily understood. This may be natural language text, or a graphical representation of the rules or facts and their relationships. The domain expert does not also have to be a software

expert in order to understand the expert system's behavior. If done well, the explanation is in a form that helps locate particular rules or facts that are incorrect.

Approach To An Air Traffic Expert System

The approach being followed is to incorporate an air traffic controller's expertise into an automated system using the expert system methodology. A feasibility demonstration is now being developed and consultations have been held with a retired controller. Some standard training scenarios have been obtained as test problems. At each of the en route centers there is a training simulator called the dynamic simulator (DYSIM) and standard sets of test problems. These problems are localized to particular air traffic environments. In other words, Boston Center has Boston Center training problems, not "National Standard" training problems, although the general nature of the problems, the types of situations that arise, and the level of difficulty are constrained by guidelines from the FAA Academy.

For purposes of a feasibility demonstration, many of the real aspects of ATC are being simplified. Concentration is on the simpler environment of high altitude en route traffic control. And in fact the initial focus is on these training scenarios, not on real traffic data. No account is being taken of weather conditions, radar outages and other less common occurrences that do appear in the training scenarios. For further development, of course, these simplifying assumptions would have to be removed.

An eventual operational version of this system could potentially be used as a controller's assistant, similar in concept to the pilot's assistant discussed in another presentation. There are a lot of operational concerns with that concept, but the operational issues are not being addressed at the moment by the MIT Lincoln Laboratory. What is being focused on are the technical issues of whether automated traffic control can be performed, how it might be implemented, and how it would behave. To this end the problem of how such a system would interface to the human controller is not now being addressed. Instead the focus is on a totally autonomous system.

It would also be possible to adapt such a system for use in training controllers. This would provide a useful service in a non-safety critical area while allowing the system development process to continue by exposure to a wide variety of traffic situations.

Potential Benefits

There are several potential benefits of an expert system approach to an automated controller's assistant. One is that the system may be more understandable to the controller. One problem with some decision aids is that they are based on mathematical procedures that are not intuitive to a controller and do not correspond to the controller's problem-solving methods. When a recommendation is made, the controller does not know how to evaluate that recommendation as it is in a different "coordinated space" from his own.

A system based on expert knowledge from air traffic controllers should behave in a way understandable to controllers and should be able to give understandable explanations. A controller could then immediately see whether proposed actions were reasonable or not. He could evaluate the basis for these actions. If he was dissatisfied, he could ask the system to explain its reasoning. This would probably be done off-line, later, in a playback mode.

Controllers have their own individual control styles. It would be desirable to have a system that could be adapted to this style. The system is going to be the controller's assistant, and he would like it to adopt his style, not enforce its own. A simplified analogy would be the calculators that can be adapted to display using a preferred number representation. Numbers can be displayed in scientific notion or fixed point, and with the decimal point in European or American format (i.e., comma versus period). This does not affect the basic functionality of the calculator, but it makes it a lot easier for a person to adapt the calculator to his preferred notation. Another potential benefit of the expert system approach may be to provide this adaptability.

Another aspect of this is that it is unlikely that the expert system development process will stop, that there will be some point where the design of the air traffic automation system is completed, never to be changed. Conditions change and systems evolve. Furthermore, there is a large component of ATC problem solving that is site dependent. It is not just that there are local map information and particular minimum en route altitudes, and so forth, that must be learned when a controller learns a new sector. The problem-solving techniques themselves depend on the particular traffic environment. Controllers learn by on-the-job training in their local environment. Furthermore, they are only certified on particualr sectors, not on all sectors in the en route center. In discussions with controllers they often mention special procedures that have evolved to deal with frequently recurring traffic problems in their sector.

In our view an air traffic expert system would continue to be adapted to local environments and controller preferences by this process of posing problems and making modifications to the knowledge base. This process would have to be constrained so that certain global safety requirements could not be violated. It is not known at this time how to accomplish that, or in general how to operationally certify such a system.

An Artificial Intelligence Development System

In order to demonstrate these ideas an AI development system (Figure 1) has been put together. There are two Symbolics Lisp machines. One has a large disk and acts as the file server. They talk to each other over an Ethernet. Each system has a monochrome display and a color display. One of the systems has an attached camera so that color screen pictures can be taken.

Why were Lisp machines purchased, and not some more conventional system? One factor was the powerful software development environment they provide. Why is it so good? There is no single most important factor. Instead, it represents the successful integration of a large number of hardware and software capabilities centered on the Lisp language and its unique features.

For our applications, it is important to have good displays. Both the color and monochrome displays have high resolution (1000 percent 900 pixels or better) and they are well supported by the software. It is very easy to develop graphics, multiple window interfaces, and menus on these systems.

Finally, Lisp is the basis for most AI programming, Lisp machines are currently the most powerful AI processors, and they are becoming the standard

AI work station. Their speed and the ability to obtain AI software from other groups were other motivations for going this way.

The Expert System Development Environment

On this equipment, a development environment for an ATC expert system (Figure 2) has been created. A set of interfaces has been provided for the expert system that make available the same information as a real controller would have: position reports from radar, flight strip data (which give the route of flight and desired cruise altitude), radio messages to and from aircraft, and interphone messages to and from adjacent sector controllers for coordination purposes. In order to allow a human expert to view what is going on, the information flowing across these interfaces is displayed on the monochrome screen. This multi-window display (Figure 3) provides menus that allow the operator to control the system's operation, and also provides three windows for displaying flight strip information, for displaying controller-pilot (radio) messages, and for entering input parameters from the keyboard. Menu selection is by means of a mouse.

The aircraft position data is shown on the color screen as a traffic situation (map-like) display, along with airways, VORs, airports, and sector boundaries. Aircraft positions are represented by a dot surrounded by a 5 nmi diameter circle to provide a distance reference. A track history (previous track positions) is provided, the length being controlled from the observer's display. One setting prevents erasure of any of the history, providing a long term map of the paths flown. Associated with each aircraft symbol is a data tag, similar to those on standard ATC displays, giving the flight identity, altitude, cleared altitude, and ground speed. The display can be zoomed in on specific areas, and airway and sector maps can be turned on and off, by means of menu items on the monochrome display.

The flight strip information is mouse-sensitive, that is, flight strips act as menu items. When the flight strip for an aircraft is selected, a first level menu appears that allows the operator to issue ATC commands to the aircraft, move the aircraft's data tag on the situation display, and delete this tag. If ATC commands are selected, a second level menu appears showing the commands currently accepted by the simulated aircraft. The following commands are currently available:

- Report aircraft heading, altitude, or airspeed
- Fly a given heading (magnetic)
- Turn left or right to a given heading (forcing a particular direction of turn)
- Turn left or right by some number of degrees
- Resume own navigation (puts the aircraft back on its flight plan after a period of vectoring)
- Climb/descend and maintain a given altitude
- Increase/reduce speed to a given value (indicated airspeed in knots)
- Increase/reduce speed by a given amount.

Figure 1. Artificial intelligence development system.

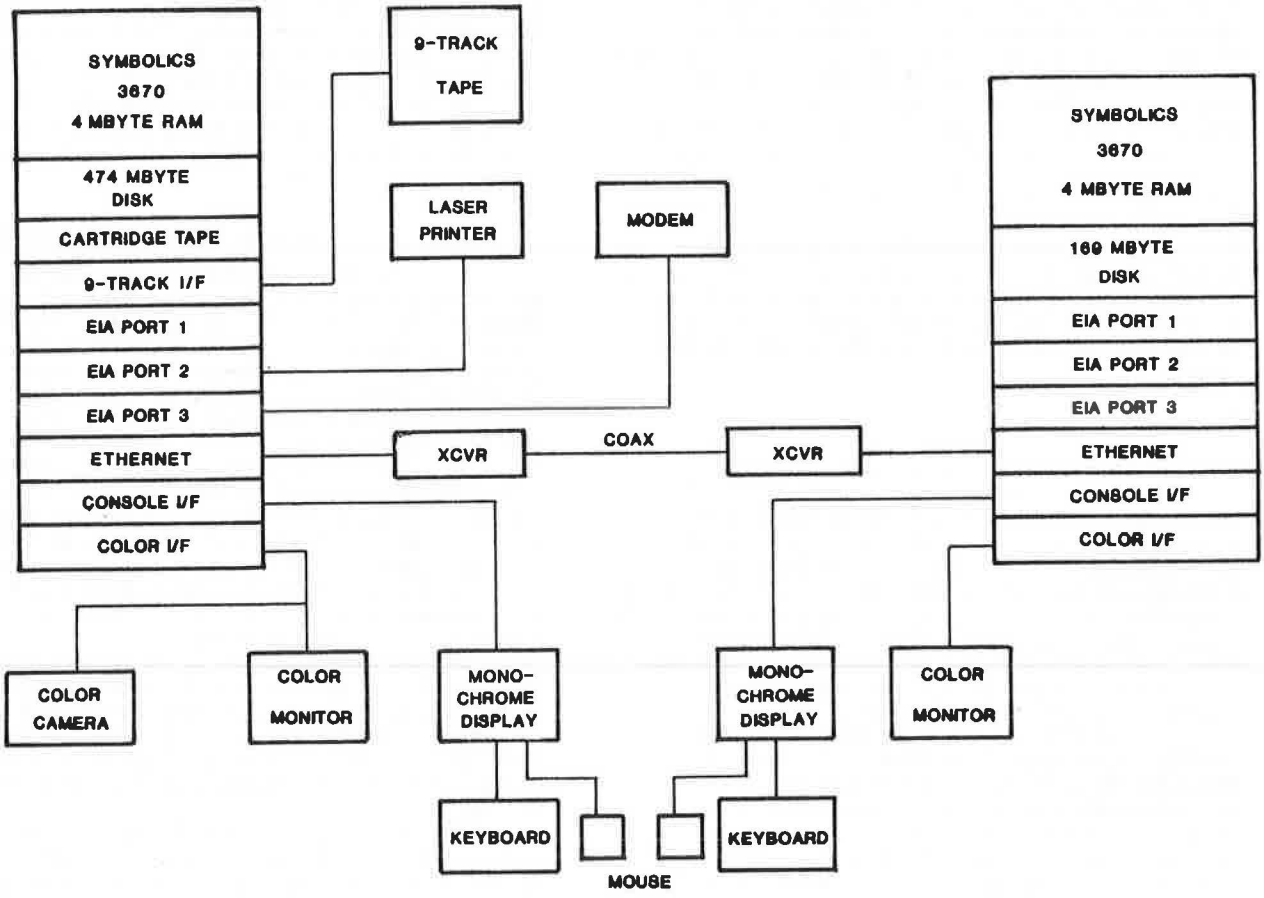
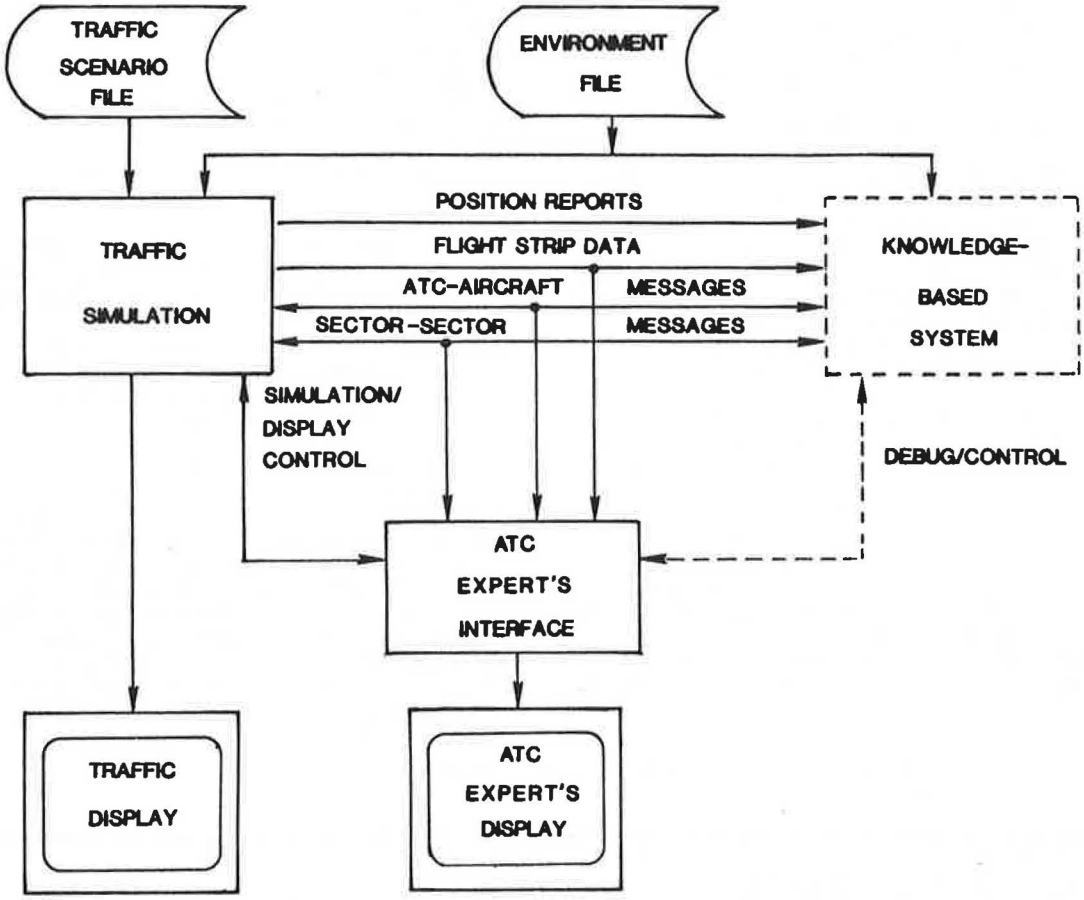


Figure 2. An ATC expert system development environment.



A major component of this environment is a traffic simulation. It is driven by a traffic scenario file which is basically the aircraft flight plan information. These data are derived from the DYSIM problems. Data have been obtained for the easiest ten (out of 18 total) training scenarios from the Boston Center. There is also an environment file which specified the map information: VORs, airports, airways and so forth.

This traffic simulation was developed using some ideas and code from a Lisp-based simulation developed at the MIT Flight Transportation Laboratory by Professor Antonio Elias and Dr. John Pararas. Work was started in September 1984 and a working version similar to the current one was available in April 1985. During the period two people worked about half time on design, coding and debug of the software, one person on the traffic situation display and the other on the simulator and the operator's display. The entire operator's interface, including the multi-pane operator's display, was implemented in two weeks in essentially the current form, using system utilities for creating multi-pane windows and menus.

Purpose Of The Development Environment

The purpose of the development environment is two-fold. First, with the knowledge-based system turned off it is possible for a person, such as a real controller, to control the simulated traffic. The necessary man-machine interface is provided by a combination of menus and text input. The controller can be observed doing this, and his behavior incorporated into rules in the knowledge-based system. Then the controller is able to criticize the performance of those rules when the knowledge-based system controls the simulated traffic. Thus, facilities are provided to first find out what the necessary ATC knowledge is and then to demonstrate that it has been correctly implemented.

A Traffic Scenario

[At this point of the presentation the first and easiest of the DYSIM scenarios was presented. The technical difficulty of reproducing color images and the loss of resolution when screened images are reduced to publication size have made it necessary to delete this portion of the presentation from the paper.]

Comments On The Scenario

In this first DYSIM training scenario the simulations that arise are of a few simple types:

1) Arriving aircraft for airports underneath the sector or in nearby sectors must be allowed to descend to appropriate altitudes.

2) Departing aircraft must be allowed to climb to their cruise altitudes, as specified in their flight plans.

3) There are a number of cases where flight paths cross. These can lead to conflicts

depending on the altitude behavior of the aircraft. However, there are no built-in conflicts where two aircraft in the scenario are co-altitude on conflicting paths. Thus, this acts more as a constraint on solutions to 1) and 2).

4) There is one instance of two arriving aircraft for Boston converging at the Albany VOR and staying together for their remaining time in the sector. Again, there is no direct conflict initially, but this complicates solutions to 1) as the aircraft will be in conflict if they are both allowed to descend to the same altitude as required by the arrival procedures. One solution involves sequencing the two aircraft.

5) There is at least one opportunity to expedite flow by giving a direct routing, possibly involving a radar vector.

It should also be noted that this is not a complete representation of this problem as it is used at the Boston Center. In addition to the basic aircraft flight paths, which are automatically simulated by both DYSIM and this environment, there is also a set of manual inputs to the DYSIM environment. These are indicated in notes that are given to the instructors running the DYSIM. In the DYSIM the instructors play the roles of simulated pilots and adjacent sector controllers, manually handling verbal communication and some decision making that is difficult or impossible to simulate automatically. This also allows them to vary the scenario from run to run. The notes for the first training scenario indicate the following complications are to arise:

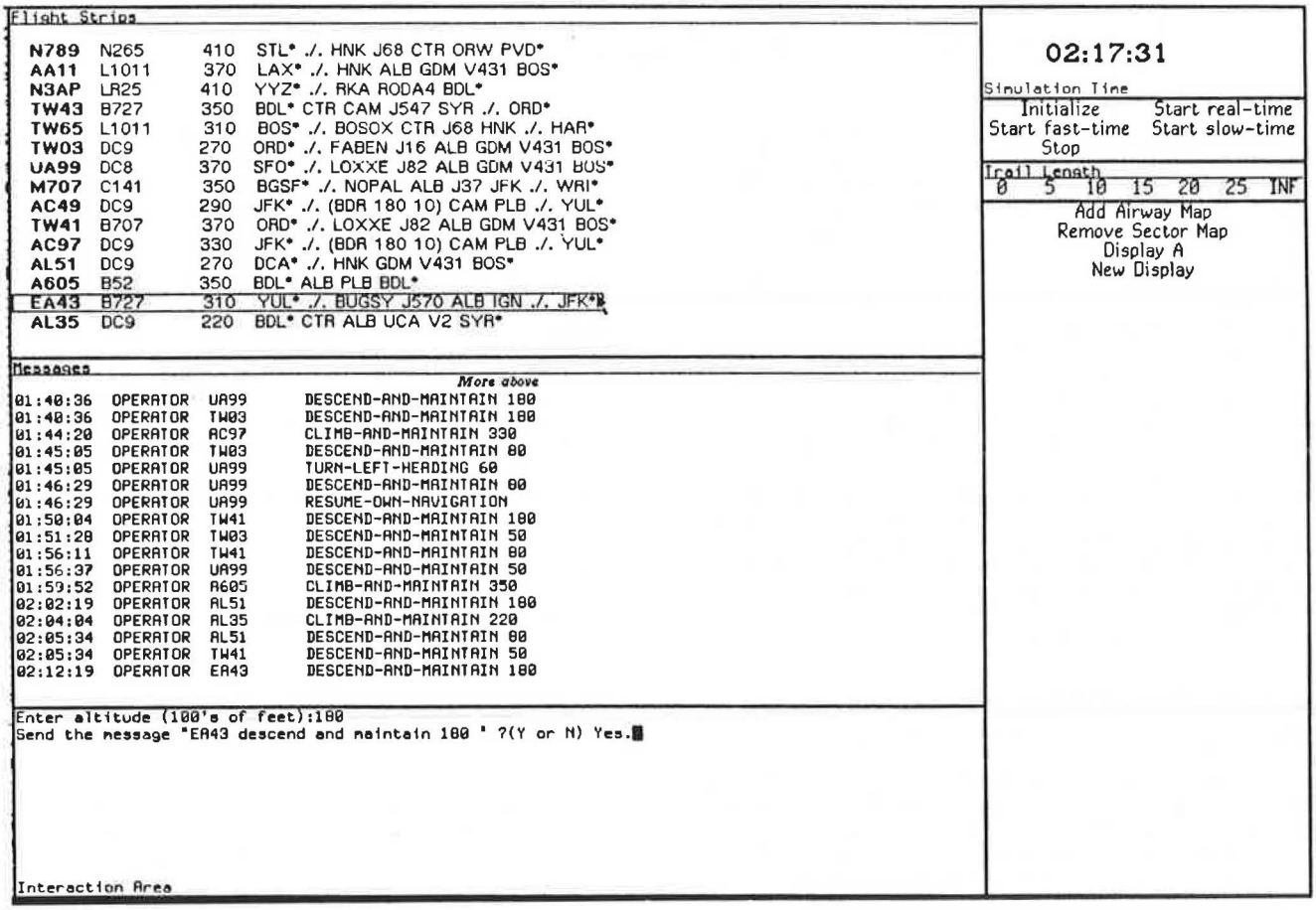
1. Two aircraft request radar vectors to fixes.

2. One aircraft requests to descend below positive control airspace, to cancel IFR, and requests traffic advisories.

3. Radar outages occur at different times in two adjacent sectors requiring re-identification of aircraft entering from those sectors.

In addition, there are the normal requirements for coordination with adjacent sectors. This can be straightforward when aircraft crossing the boundary are in radar contact and on their flight plan route. It is more complex if the adjacent sector has had a radar failure, or if an aircraft has been vectored off its route or is otherwise not conforming to the standard procedures. The development environment does not currently support any of these activities, although a rudimentary form of handoff will be added soon.

Figure 3. Monochrome display of an AI traffic scenario system showing menus, flight strip information and controller-pilot messages.



Conclusion

In conclusion, the author emphasizes the following points:

- The expert system development methodology requires an extensive set of sample problems. It encourages system developers to test their ideas against these problems early in the development life cycle.
- In the ATC setting, this requires a simulated traffic environment, possibly augmented by real-time human inputs in areas difficult to simulate.
- The current simulation is adequate for initial expert system development. Some additions will be needed to demonstrate all aspects of the first training problem.
- Continued development of an air traffic controller expert system will require continued development of the simulation to add to its functionality and to improve the fidelity with which it simulates those functions.

Discussion

George E. Swetnam, Mitre Corporation Does your simulation do anything to help present the conflicts to the controller? What you have done essentially is to replicate the information that is available to him presently on the traffic display. Has any thought been given toward showing him the conflicts in some other form that will help him grasp what the expert system is doing?

David A. Spencer Not particularly. The nice thing about this is that it is a simulator, and you can stop it. If somebody wants to analyze the situation we can stop it at some particular point and look at it, for as long as we want. We do not have to instantly present complex graphics so that a real-time decision can be made.

FAA Comment If I can add to that. Right now we are trying to see if we can get the system to work, but you are exactly right. There are major human factors issues, not only in presenting the conflict but presenting the resolution. These kinds of issues are very crudely understood.

David A. Spencer We are basically working with the problem-solving aspect of air traffic control, defining the problems, defining the solutions, and discovering how to generate reasonable solutions. How to present these to the controller and how he is to use them are complex questions that we are not equipped to handle at this time.

Robert H. Brown, NASA Johnson Space Center Two questions. How many rules do you have, and are you bothered with garbage collection?

David A. Spencer The second one is easier. No, because we just use a large virtual memory.

Robert H. Brown, NASA Johnson Space Center You are using the Symbolics machine?

David A. Spencer We are using the Symbolics 3670.

Robert H. Brown, NASA Johnson Space Center You can run it long enough?

David A. Spencer You can run it long enough, yes. It can run through an entire simulation without garbage collecting.

Robert H. Brown, NASA Johnson Space Center How long is that?

David A. Spencer An hour of real-time, 15 minutes if you run it in fast time mode.

Robert H. Brown, NASA Johnson Space Center How many rules?

David A. Spencer We have not really gotten into the expert system. The expert system portion was shown dotted for a reason. At the time the display was made it did not exist. At this point in time it is a shell. The interfaces have been put in. We are now implementing some rules for finding path intersections, lines and line intersection points, that sort of thing. It is at that level at this point. We do not have rules that actually implement air traffic control.

Robert H. Brown, NASA Johnson Space Center Do you have an estimate of how many rules?

David A. Spencer There is some feel for that. Several people, some here, in fact, have demonstrated that a relatively small number of rules can handle surprisingly complex cases, on the order of 20 to 30 rules.

Curtis A. Shively, Mitre Corporation We have 100 rules.

EXPERT SYSTEM FOR DOPPLER WEATHER RADAR INTERPRETATION

Steven D. Campbell
Massachusetts Institute of Technology
Lincoln Laboratory

Overview

This presentation concerns an expert system being developed for Doppler weather radar interpretation. The objective is to use artificial intelligence (AI) techniques to interpret weather radar displays to recognize wind shear hazards. The reason for developing an automatic recognition capability is that terminal Doppler weather radars will be placed at many airports to detect these hazards and it is not cost effective to put expert radar meteorologists at each of these locations. Thus, an automatic recognition capability is desired which can place a warning on the air traffic controller's screen so that these hazards can be avoided.

The approach being taken is to capture the expertise of a radar meteorologist in recognizing these hazards. Radar meteorologists exist who are very good at picking out microbursts from Doppler radar displays. The goal of this project is to understand what their expertise is, and to try to build it into a computer program so that it can be replicated at many sites.

This presentation will discuss expert systems briefly, summarize the characteristics of wind shear hazards and Doppler radar, outline the design

of the weather interpretation system, and finally present some initial results.

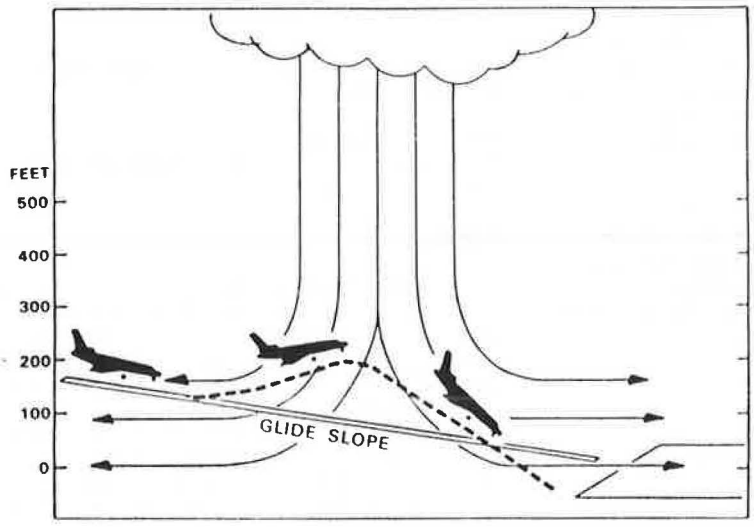
Rule-Based Expert Systems

Rule-based expert systems have been built to perform expert level tasks in a number of different domains: medical diagnosis, VAX computer configuration, geological data analysis, and a number of other areas. These systems consist of a set of production rules in the form of condition - action (IF-THEN or antecedent-consequent) pairs, and these encode the heuristic knowledge of the system. There is a working memory that contains known facts about the situation, and an inference engine that matches those facts against the condition (IF) part of the rules. When the condition part of a rule is satisfied, then the action part is carried out.

Wind Shear Hazards

Wind shear is a change in wind velocity over some distance. When this change in velocity is large over a small distance, a wind shear hazard results. There are two kinds of wind shear hazards of primary interest here - microbursts and gust fronts. Microbursts are known to have caused crashes at New Orleans and John F. Kennedy Airports, and possibly the recent event at Dallas International Airport. Gust fronts are less hazardous than microbursts, but have a major impact on runway operations at airports because they are associated with wind shifts. The ability to anticipate such wind shifts and assess what kind of impact they are going to have on an airport's operations is very important.

Figure 1. Aircraft encounter with a microburst.



Microbursts

In a microburst there is a very strong downdraft which spreads out at the surface, and this can pose a problem for aircraft. For example, in Figure 1 the aircraft is on the glide slope and gets into a region of strong head wind as it enters the microburst. This causes the aircraft to gain lift, hence go above the glide slope. The pilot typically will try to correct for this by putting the nose down and/or reducing thrust. The aircraft then gets into a downdraft and starts losing altitude. Finally, it gets into a tailwind and loses airspeed and lift. This can cause the plane to crash short of the runway. Similar effects can also cause a crash on takeoff, as happened in the New Orleans crash.

Microbursts are very short-lived events, typically lasting on the order of 5 to 10 minutes (Figure 2). They begin in the upper atmosphere and descend to the surface where they spread out. About five minutes after their onset at the surface the most severe winds occur; they then dissipate and are usually over in ten minutes. The spatial scale of these events is on the order of 4 kilometers, at least for the initial outflow.

A model of at least one type of microburst has been proposed by Fujita of the University of Illinois and is shown in Figure 3. It involves an inflow or convergence of winds at upper altitudes, a downflow which may be rotating and then a surface outflow. This is the type of meteorological model that the knowledge base of our system needs to capture.

Doppler Weather Radar

The relevant characteristics of Doppler weather radar will now be summarized. Primary products are reflectivity, which measures rainfall rate, radial velocity, which measures the component of the wind velocity along the radar beam, and spectrum width, which is an indication of turbulence. There are also some derived products. One of them is radial shear, which is the derivative of the velocity taken along the beam and is an indication of inflow or outflow. Azimuthal shear is the derivative of the radial velocity, but taken in the azimuthal or cross-beam direction. Azimuthal shear can be used to detect rotation.

This data is collected in a set of constant elevation angle scans which are called tilts (Figure 4). A volume scan consists of several tilts which start at low elevation, then step up to successively higher elevation angles. For the terminal Doppler weather radar environment, a volume scan takes about two minutes.

Figure 2. Microburst life cycle.

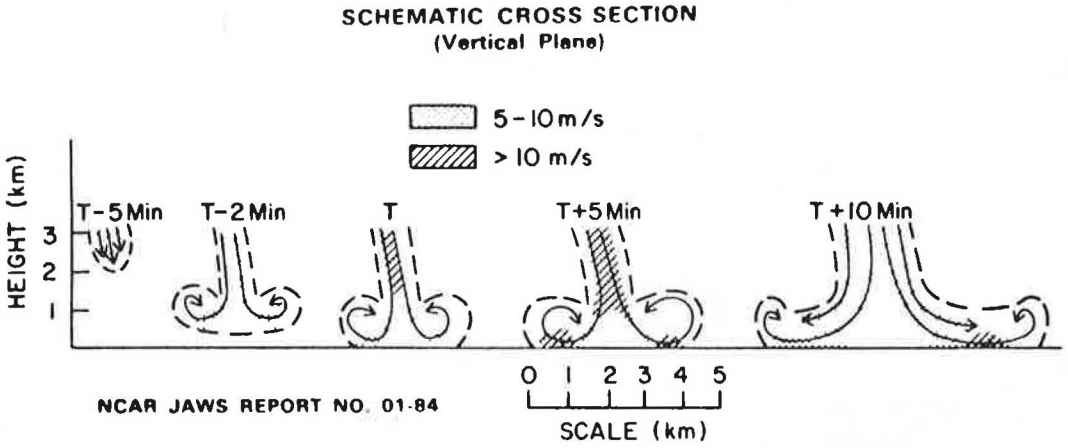


Figure 3. Model of a surface microburst.

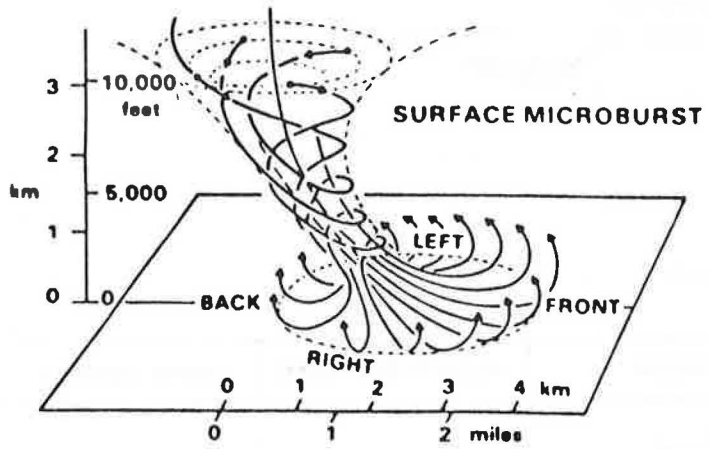


Figure 4. Single radar volume scan: A collection of tilts at various elevations.

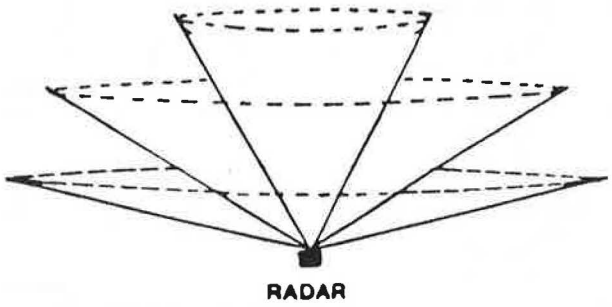


Figure 5. Primary signature of a microburst.

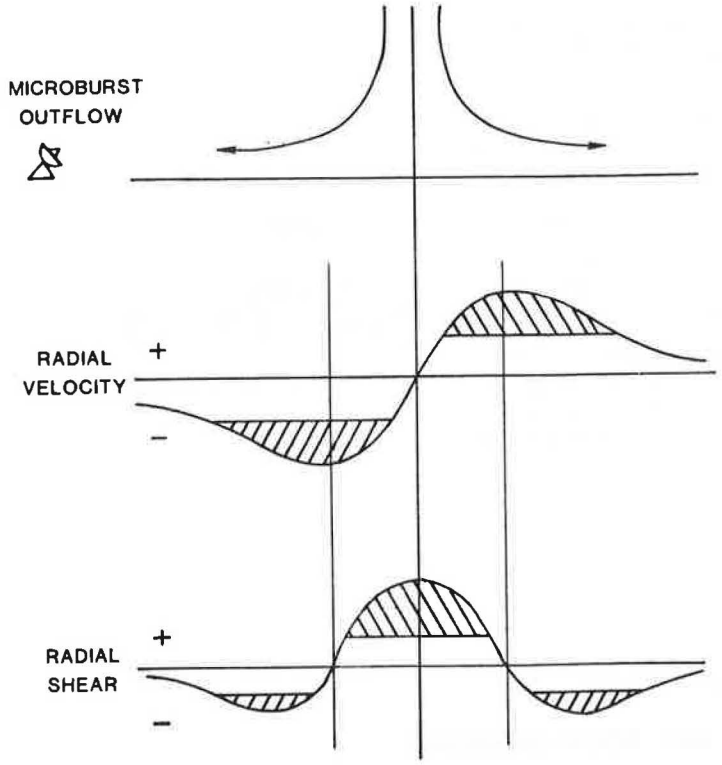


Figure 6. Summary of single-Doppler radar signatures.

	FLOW FIELDS	VELOCITY SIGNATURES	SHEAR SIGNATURES
INFLOW	RADAR	+ · −	+ − + RADIAL
ROTATION	RADAR	− / +	− / + / − AZIMUTHAL
OUTFLOW	RADAR	− · +	− + − RADIAL

Wind Shear Signatures

The primary signature of a microburst is the surface divergence or outflow. This outflow creates a flow toward the radar (negative radial velocity) on the side nearer the radar and a flow away from the radar further out (Figure 5). If the derivative of radial velocity is taken along the radial, the resulting radial shear is first negative, then positive and then negative again. This region of positive radial shear is an indication of a strong divergence area or region of outflow.

A similar sort of analysis for inflow (Figure 6) shows first a positive velocity region and then a negative velocity region, resulting in a negative radial shear region. For rotations, the result is a velocity couplet which is oriented at right angles to the radar beam. This is where the azimuthal shear is used. As the radar beam moves clockwise, it goes from a region of negative velocity to a region of positive velocity. This gives positive azimuthal shear.

What is wanted is to tie the model of a microburst to the radar observables (Figure 7). The model shown here for a surface microburst has surface divergence, middle-level rotation and an upper level convergence. The signature for a surface divergence could be a velocity couplet or it could be a radial shear or it could be both. Similarly for rotation it could be velocity couplet or positive azimuthal shear region. For convergence it could be a negative radial shear region or a velocity couplet.

System Design

The first generation system now being developed is intended to recognize a limited set of weather hazards, microbursts and gust fronts, in non-real time. The approach is to couple a rule-based expert system to a powerful image processing package, since this particular application has a very high visual processing component.

The basic system design is shown in Figure 8. There are two components, an observer component and an expert component. It is as if there was an expert meteorologist in one room who cannot see the radar displays, and a naive observer in another who can see the displays but does not have meteorological expertise. The expert system sends queries to the observer asking about the radar data and about the features that have been extracted from it by the feature processing system. The responses by the observer are processed by the expert system, and a symbolic representation of the radar data is built up in the working memory. The expert system then operates on that symbolic representation using the production rules to recognize wind shear hazards.

The basic processing flow of the system is shown in Figure 9. First a radar data base is built up. This data base is a structured data base consisting of a set of volume scans. Each volume scan is composed of tilts, each tilt is composed of a set of radar products (reflectivity, velocity and so forth, plus derived products) and each product consists of a set of Cartesian resampled pixels.

Figure 7. Model of Denver microburst.

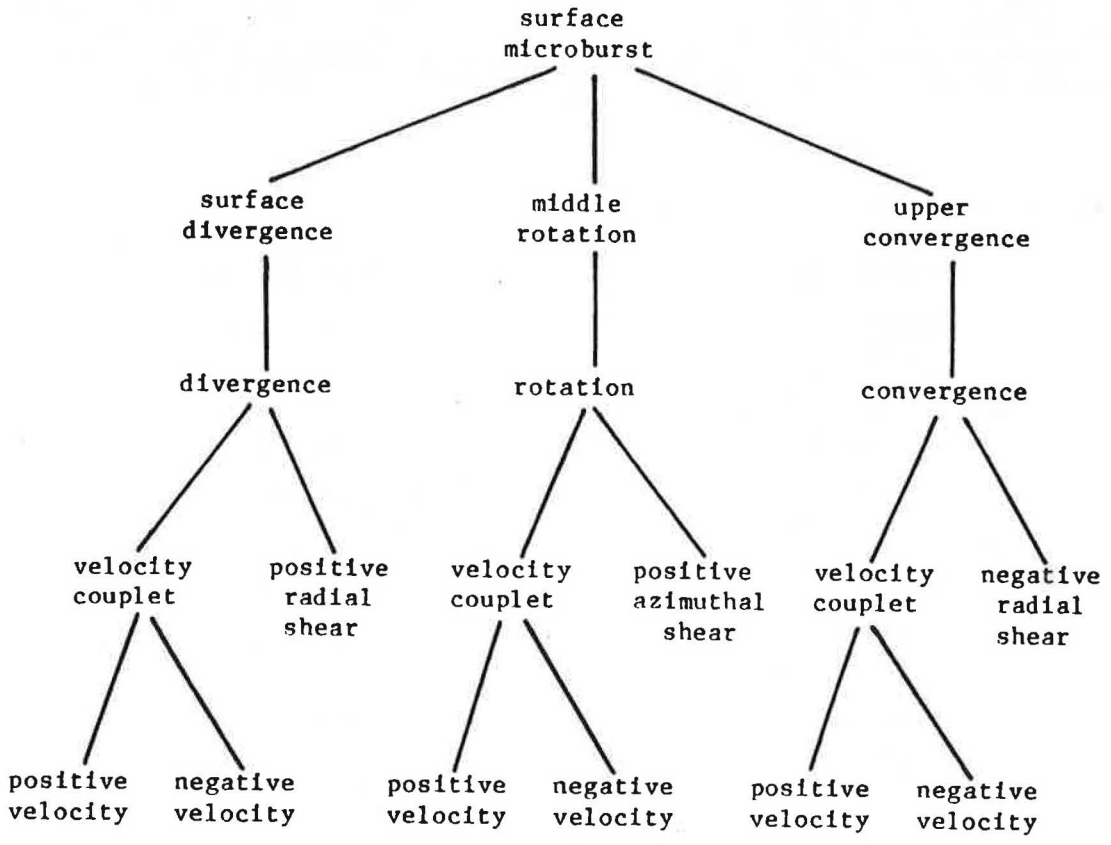
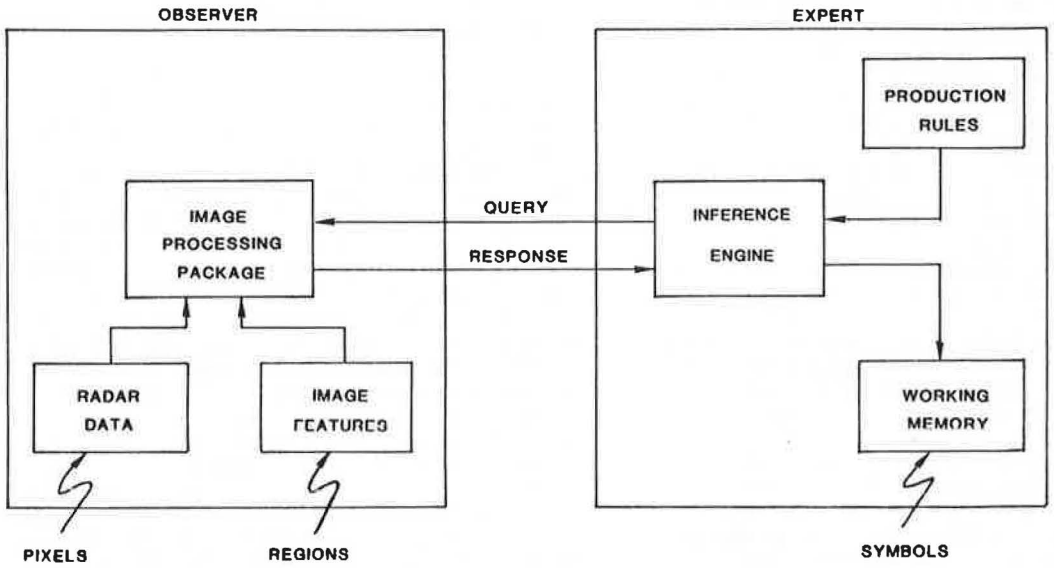


Figure 8. WX1 system design.



The next step is a feature extraction process. First, the pixels are thresholded to form classes, such as positive velocity, negative velocity, positive azimuthal shear and so forth. Next, connected regions are found and labeled as features. Then the program starts building up what we call a feature data base in which features are assembled into more complex features. For example, putting a positive velocity feature and a negative velocity feature together to form a velocity couplet creates a tilt feature. Tilt features are combined to create volume features such as microbursts. An example of a data set feature would be a microburst that is recognized over several successive volume scans.

Figure 10 shows an example of a feature as it is represented in an object-oriented programming system on the Lisp machine. This is a connected region of a particular class, in this case a positive velocity feature. This feature has instance variables attached to it, such as the feature label, class, number of pixels, etc. Also computed is the bounding box, which is defined by the maximum and minimum X and Y values of the object. This is very useful in expediting processing. There are also many methods (message handlers) which are attached to features. Messages can be sent to a particular feature to ask it, for example, to return its centroid, its size, its length, and so forth.

Figure 9. Basic processing flow of WX1 system design.

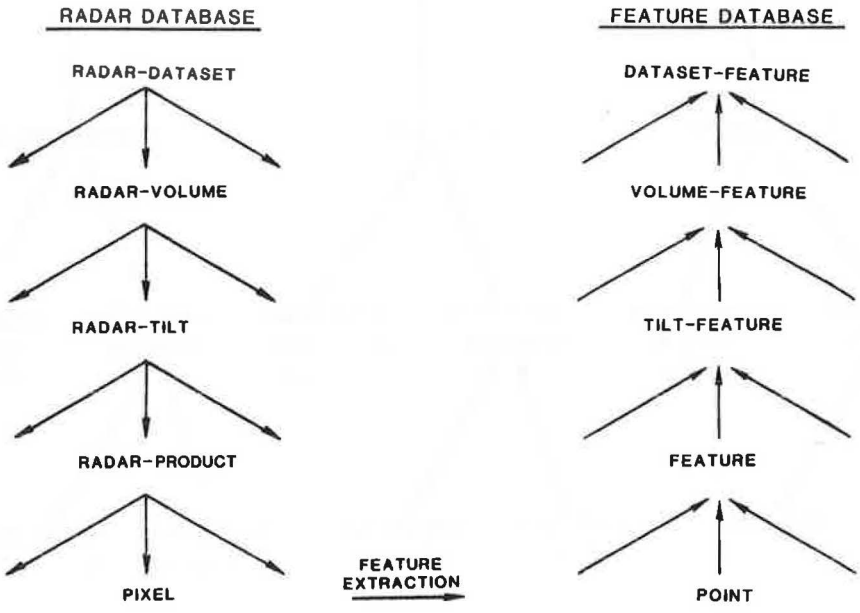
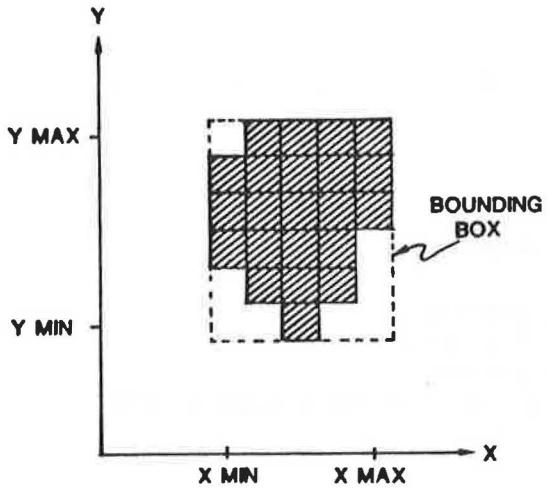


Figure 10. Example of a feature represented on the LISP machine.




```
#<FEATURE 1>, an object of flavor FEATURE,
 has instance variable values:

 RADAR-FIELD: #<RADAR-FIELD VELOCITY 1>
 FEATURE-NUMBER: 1
 FEATURE-SIZE:   30
 FEATURE-CLASS:  :POSITIVE-VELOCITY
 XMIN: 162
 XMAX: 166
 YMIN: 130
 YMAX: 135
```

Figure 11. An expert system's radar feature data base as represented in working memory.

The linkage between the radar feature data base and the symbolic representation in the production system's working memory will now be described. Suppose there is an input field, say a velocity field V_1, and the feature extraction process has been performed on it (Figure 11). Suppose that positive and negative velocity features F1 and F2 are extracted from V_1. These features are represented as facts in working memory. Feature F1 is a positive velocity feature. Feature F2 is a negative velocity feature. The expert system does not know any of the details of a feature, just that it is a region of a certain class. But it can ask questions about that particular feature. Basically the expert system knows there is a blob out there, and it can ask questions about the blob.

Figure 12 shows an English language representation of rules for processing such a velocity field. After completing feature extraction, the system does an evaluation of the raw features and decides which ones are likely to be of interest (see second rule in Figure 12). Call these candidate features.

Now suppose F1 and F2 satisfy the conditions to become candidate features, a positive velocity feature and a negative velocity feature. The third rule in Figure 12 will match those features, with the variable FP matching F1 and the variable FN matching F2. The rule now tests that the distance between the two is less than 4 kilometers, and that the difference in velocity between the two is greater than 10 meters per second. If the tests are satisfied then a velocity couplet fact is created by the action part of the rule.

The creation of that velocity couplet fact triggers another rule (the fourth rule in Figure 12). The rule says that if a velocity couplet has been found and the orientation is appropriate, then label it as a divergence signature. The next rule asks

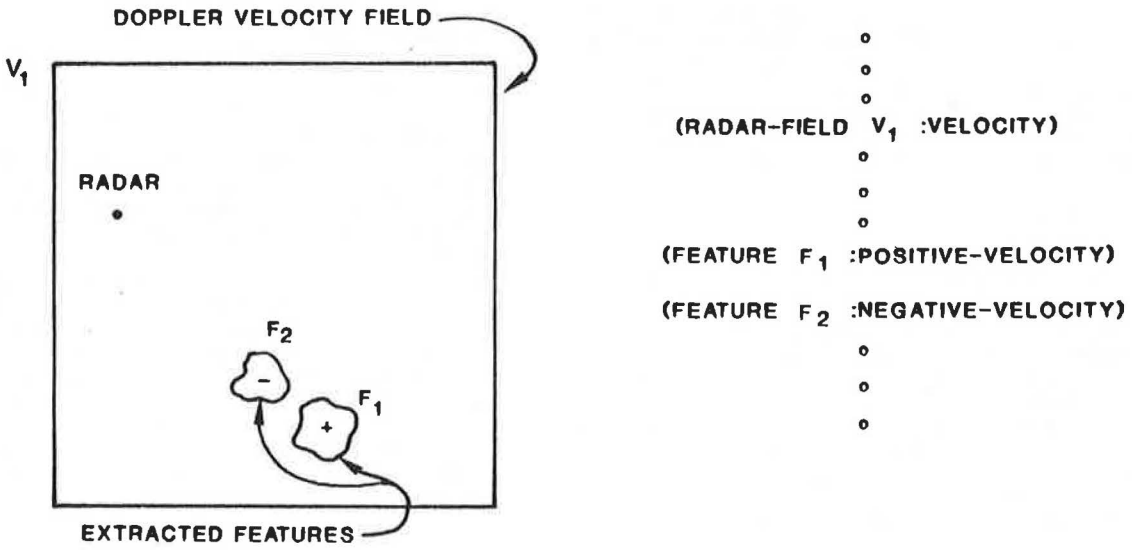


Figure 12. An English language representation of rules for processing a working memory's velocity field.

```
( If    VF is a velocity field
  Then  extract regions from VF
        for each region, create unevaluated velocity feature )

( If    FV is an unevaluated velocity feature
        size of FV is less than 4.0 km
        shape of FV is compact and not elongated
  Then change FV to a candidate velocity feature )

( If    FP is a candidate positive-velocity feature
        FN is a candidate negative-velocity feature
        distance from FP to FN is less than 4.0 km
        difference in velocity between FP and FN is greater than 10 m/s
  Then create a velocity-couplet fact from FP and FN )

( If    VC is a velocity-couplet
        orientation of VC is less than 45 degrees w.r.t. radar beam
  Then create a divergence-signature fact from VC )

( If    DS is divergence-signature
        altitude of DS is less than 1.0 km
  Then create surface-divergence fact from DS )

( If    SD is a surface-divergence
        MR is a middle-level-downdraft
        UC is a upper-level-convergence
        overlap exists between SD and MR
        overlap exists between MR and UC
  Then create surface-microburst fact from SD, MR and UC )
```

whether the divergence signature has an appropriate altitude. If it is at the surface, then a surface divergence fact is created. Finally, this line of reasoning is put together with other lines of reasoning (not shown) in a rule which says that if surface divergence, middle-level downdraft, and upper-level convergence are present, and the appropriate overlap occurs between these features, then create a surface microburst fact from those lower level features. In this way low-level features are built up into high-level features.

Initial Results

With that introduction to the methods being used, some initial results will now be presented. The original radar data for these examples has been obtained from a number of sources. One is the National Center for Atmospheric Research (NCAR) in Boulder, Colorado. Two projects were done there, the JAWS project in 1982 and the CLAWS project in 1984. These were primarily projects to gather data on microbursts. Gust front data was obtained from the National Severe Storms Laboratory (NSSL) in Norman, Oklahoma. Finally, the MIT Lincoln Laboratory has a terminal Doppler radar program with a test radar in Memphis, Tennessee.

Microburst Example

This example shows the system's performance on one set of microburst data taken near Denver. Figure 13 shows the result after the system has performed feature extraction on the radar data [the raw radar data has not been shown due to technical difficulty in reproducing the color images]. It produces a set of candidate features which are shown in Figure 13. In the low elevation tilt, Tilt 1, it finds two big radial shear features which are likely microburst outflows. It has put a box around each. It also finds one velocity couplet region which is appropriately oriented to be a surface outflow.

In Tilt 2 it does not find the velocity couplet for the rotation because it is so asymmetric, but it does find an azimuthal shear region or region of rotation. The system relies on the overlap of the surface divergence features with the middle altitude rotation feature, and declares that a microburst exists at that location (Figure 14).

Figure 15 shows the result of processing some NCAR data where a sequence of seven volume scans was available. On four of those volume scans the system found a microburst. There were actually two microbursts. One microburst was recognized on two successive volume scans, and the system correctly

Figure 13. Candidate microburst features detected in Denver radar data.

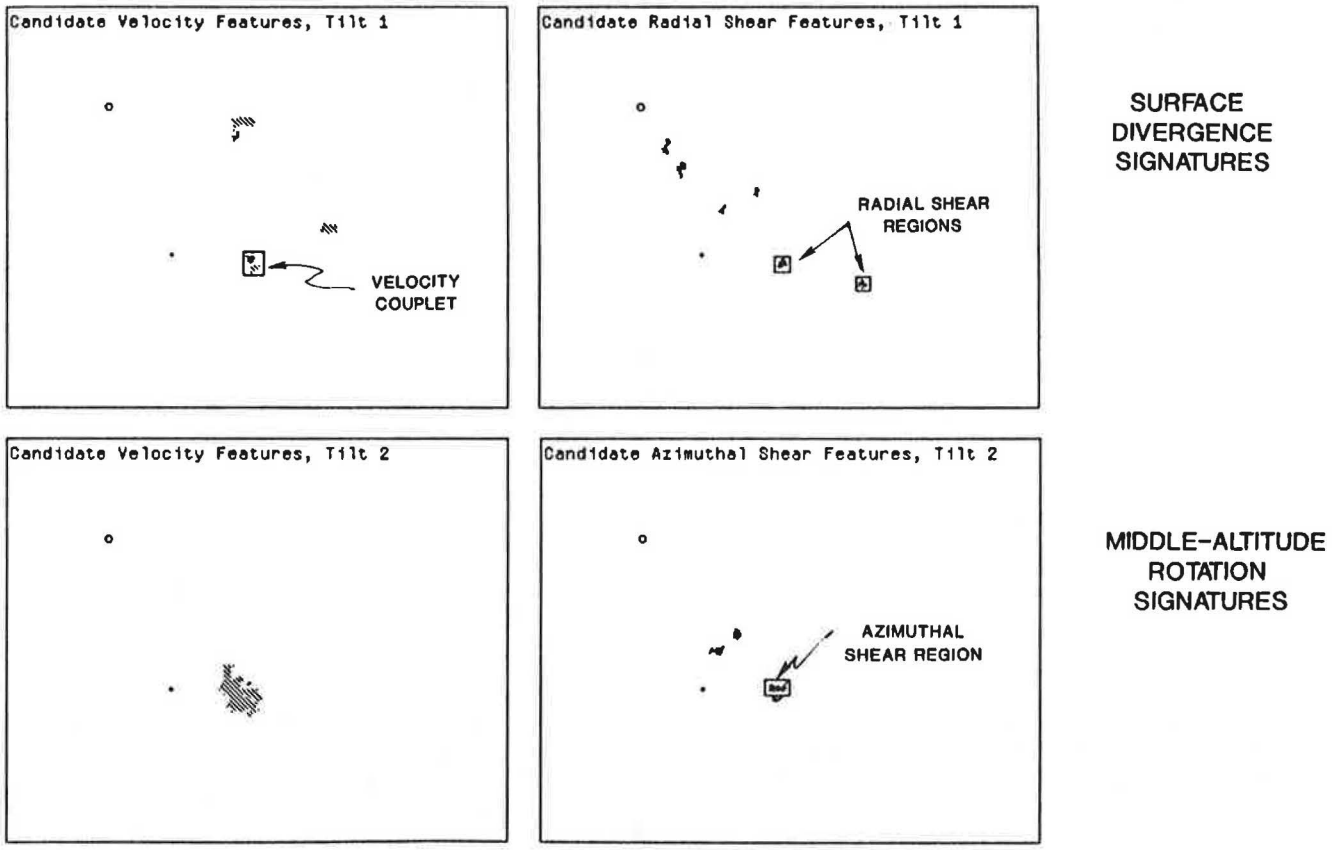


Figure 14. Recognized surface microburst features in Denver radar data.

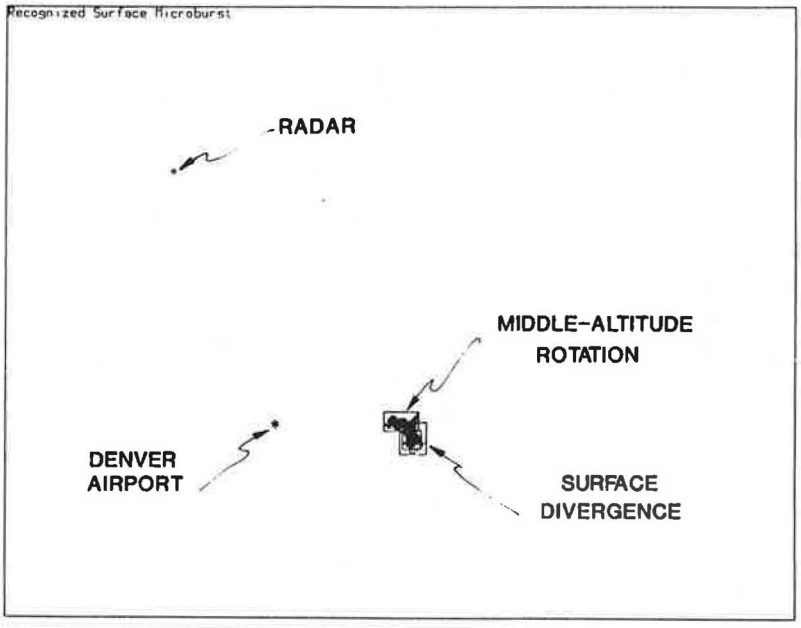
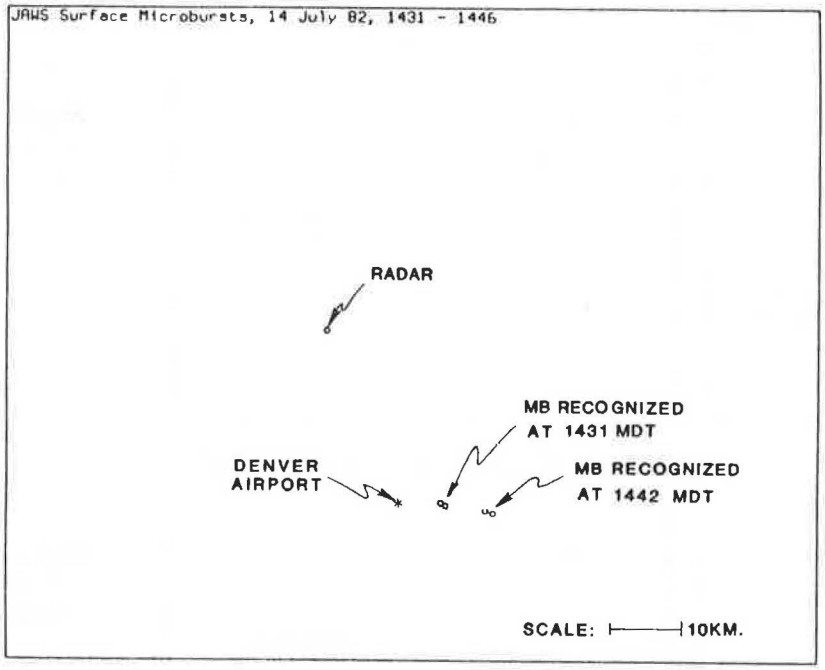


Figure 15. Additional microbursts detected in Denver radar data.



put them together as the same microburst. Subsequently it found another microburst on two other volume scans. These results have been confirmed from published reports.

Gust Front Recognition

Figure 16 is a diagram showing the origin of gust fronts. In a convective storm there is a downdraft and a region of cold air flows out of the storm. This cold air outflow running into a warm air inflow creates a region of turbulence called a gust front.

Figure 17 shows this more clearly. At the top of the figure is shown the cold air outflow colliding with the warm air inflow. A line of convergent airflow results, in contrast to the outflow caused by a microburst, and there is therefore a region of negative radial shear. The rule system looks for this region of convergence. It expects to find long, thin regions of negative radial shear.

One of the aspects of the gust front recognition system is that it is able to put together features that may become fragmented during the feature extraction process. In the hypothetical example in Figure 18 a long thin region of shear was found in Tilt 1. On the next tilt up, two separate such regions are found. The system is capable enough to go back, knowing that there was an overlapping line of convergence found on another tilt, and merge these two pieces together into one shear line.

Gust Front Tracking

Figure 19 shows the results of processing some data from the National Severe Storms Laboratory. The raw radar data has again been omitted due to the technical difficulty of reproducing color images. A squall is propagating to the east, and there is a shear line due to its gust front. About 15 minutes later it has moved to the east some distance. Four volume scans of data were available, and this figure shows a schematic representation of the results. The system was programmed to draw a line for each shear line found and to mark the centroid of that line. There is a clear propagation of the gust front to the east, and for this it is possible to estimate the propagation speed. Work is now going on to track gust fronts and predict where they will be in successive volume scans. This will be very useful in assisting air traffic control in anticipating wind shifts.

Summary

To summarize, a first generation system to interpret Doppler weather radar data is under development. It employs an expert system, coupled with a powerful image processing capability, and rules are being developed for detecting low altitude wind shear hazards. At the moment the microburst algorithm runs about six times slower than real time, and the gust front algorithm runs about four times slower than real time. The image processing calculations are the primary limitation on processing speed.

Discussion

Question Regarding the signature that you described before on the microburst, would that be the same for all over the country with the different microbursts?

Steven D Campbell No, they do not have the same structure. It turns out that microbursts vary quite a bit from place to place. Memphis, Tennessee microbursts always have a lot of rain. Microbursts in Denver usually have very little rain. When we started out I thought it was very simple -- convergence, rotation, divergence. We are finding out that it is not that simple. We just keep having to add more and more rules, and more and more ways

Figure 16. Aircraft encounter with a gust front.

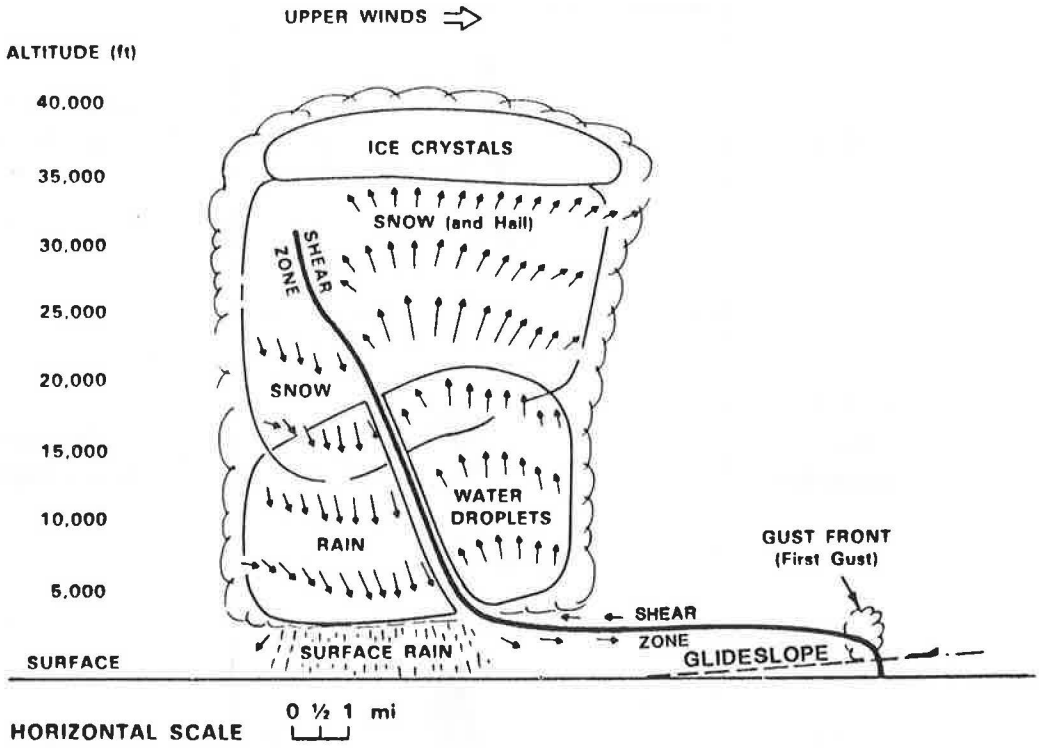


Figure 17. Gust front convergence signature.

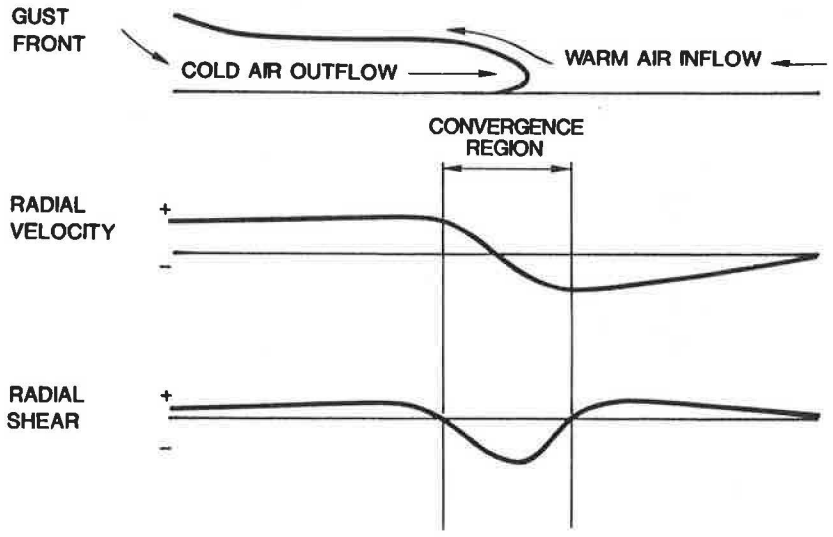


Figure 18. Merging fragmented shear features.

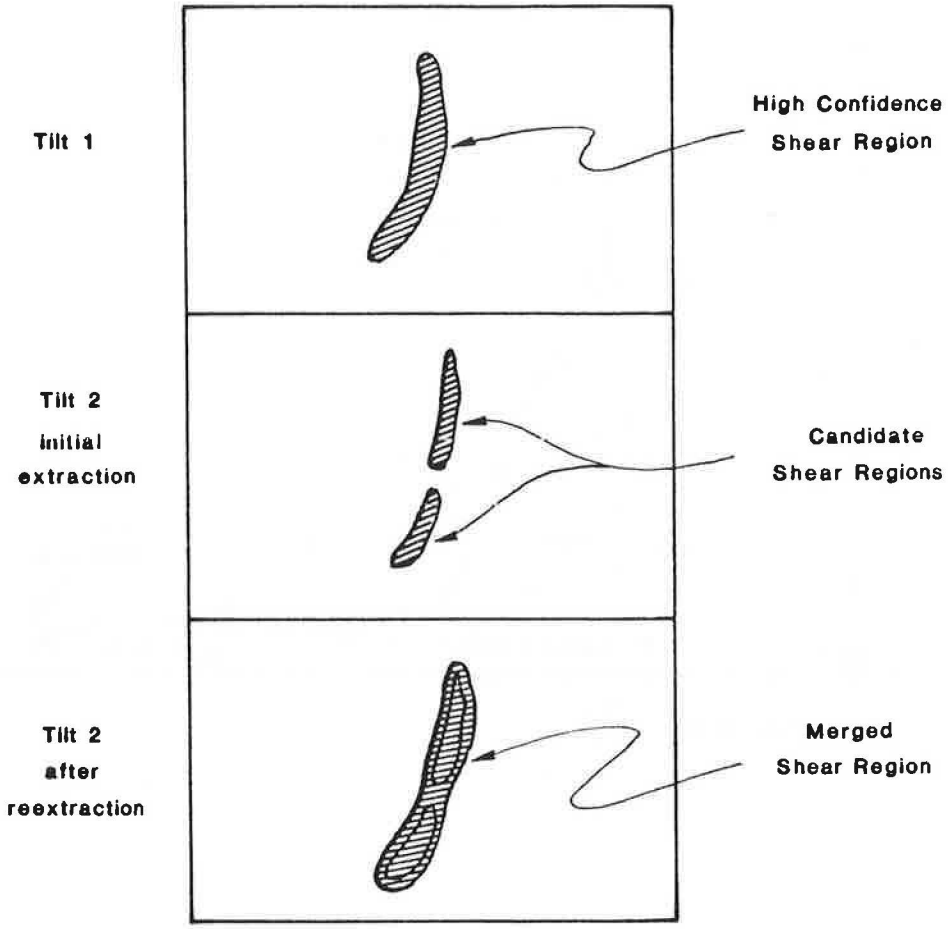
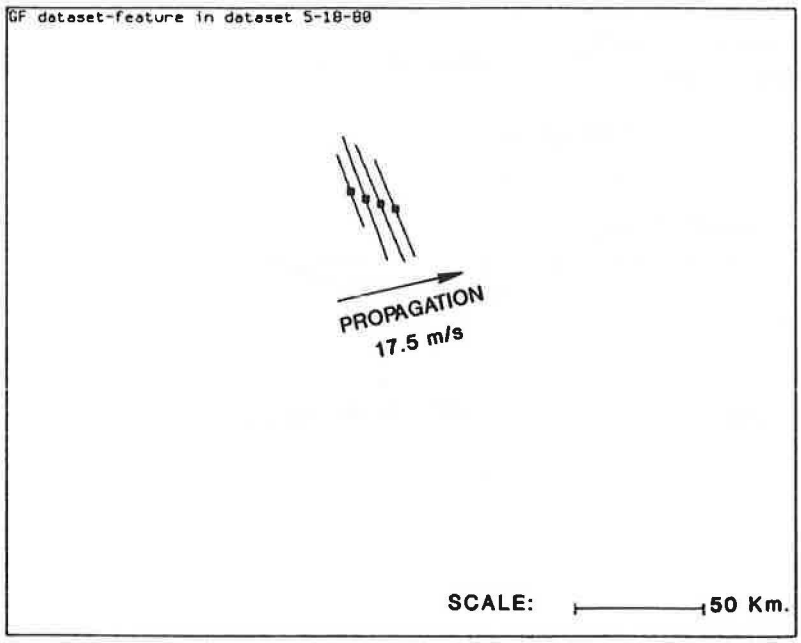


Figure 19. Gust front propagation.



of detecting different types of microbursts. Even in Denver, for example, there are a number of different kinds of microbursts. Some have a lot of rain. Some have almost none. Some have rotation, and some do not, and so forth.

Question The one that you described, a Memphis microburst. Which one would that be related to?

Steven D. Campbell That one had very heavy rain. The microburst described before, the basis for the microburst model described in the presentation, and the example in Figures 13 to 15, was the dry microburst, with almost no rain at the surface.

Question How can the radar pick it up if there is no rainfall?

Steven D. Campbell You don't need to have surface rain to pick it up. You just need water particles in the air. Those particles are not necessarily enough to cause rain at the surface. There just has to be some water, or perhaps other airborne particles.

Question Since there are so many different kinds of microbursts, how would you be able to detect all types or would the rule-based system be designed just for that particular area of the country?

Steven D. Campbell There might be some changes for different parts of the country, but basically the answer is that you add rules to take care of various different types of situations. The model that was discussed can be augmented, particularly in the downdraft area. It could be cyclonic rotation, counterclockwise. It could be anti-cyclonic rotation, clockwise. It could be a strong region of reflectivity. It could be many things. The system could just use the upper level convergence. The thing about the rule-based system is that we can add to our existing body of knowledge to cover more cases. We start off with some fairly simple assumptions, find out where these do not work, and add some more rules.

Question Wouldn't that slow down your real time interpretation of it?

Steven D. Campbell Yes, but it turns out that we are not being limited by the expert system, but by the image processing.

Question That is the question I was going to ask. As you consider more possibilities you extend the rule base. If there are more possible candidates being developed, will there be more false alarms?

Steven D. Campbell We have not had any false alarms so far. This system is very conservative, and we have not had a problem with false alarms. I think it is because of our knowledge-based approach. I think that false alarms are going to be a real problem for some of the strictly algorithmic approaches that are currently being developed.

GROUP DISCUSSIONS

Following the presentations, the workshop participants divided into two groups. The purpose of these groups was to discuss and analyze the material and information which had been presented, and to attempt to reach conclusions regarding the elements of the air traffic control system which would benefit the most from the application of artificial intelligence techniques. A preliminary report of each group's discussion was prepared, and then at the concluding plenary session of the workshop final reports were presented and discussed. These reports follow.

PRELIMINARY REPORT - GROUP 1

Ronald L. Larsen, University of Maryland

The panel was one of two which met to deliberate on the material presented at the meeting. The panel sought to:

1. Identify critical functions of the air traffic control system that can benefit substantially from the introduction of artificial intelligence techniques;
2. Identify topics in artificial intelligence that will require specific attention in order to satisfactorily address the requirements of automating air traffic control functions; and
3. Suggest realistic guidelines for developing demonstrations of artificial intelligence applications in the air traffic control (ATC) environment.

Functional Areas of Opportunity

The panel considered eight functional areas of the ATC system in evaluating high benefit opportunities for applying artificial intelligence:

1. Flight plan generation
2. Real time conflict resolution
3. Severe weather detection
4. Training aids
5. Maintenance aids
6. Flow control (traffic management)
7. Failure management
8. Dynamic separation.

Each of these topics was briefly discussed, noting, in particular, where the functional areas displayed attributes compatable with current knowledge in engineering technology. The panel sought to identify applications for which reasonably codified knowledge exists and is exercised by recognized experts in air traffic control. Tightly bounded problem domains were also considered important to successful application of current artificial intelligence techniques. Third, the panel sought application areas where successful application would yield very substantial benefits.

Four of the eight functional areas were identified as major opportunity areas for artificial intelligence based on the three criteria:

1. Flight plan generation
2. Real time conflict resolution
3. Severe weather detection
4. Flow control (traffic management).

Consideration of these functional areas with respect to the supporting artificial intelligence (AI) techniques resulted in a list of eight AI technologies that are fundamental to successful ATC applications:

A. Software verification and validation
B. Human to machine interface
C. Cooperating expert systems with distributed knowledge bases
D. Planning (including geometric and temporal reasoning)
E. Information extraction
F. Competitive goal interaction
G. ATC-oriented heuristics
H. Flight-rating the technology for deployment.

The following matrix suggests the nature of the interaction between the technology areas and the applications (Figure 1).

Figure 1. Interaction between AI technology areas and the ATC applications.

	Flt. Plan	Conf. Res.	Severe WX	Flow Cntl
V&V	plan quality	costly errors	no false alrm	reliability
I/F	crew-friendly	info urgency	stress	ease of use
C.E.S.		multi-source	corroboration	cooperation
Plan	3-D + time	quick response		economies
I-Ext.		perceive sit'n	signal proc.	radar proc.
G-Int.	WX/fuel/time	A/C-gnd prty's	risk/safety	A/C-gnd prty
Heur.		FAA guidelines	noisy data	FAA guides
Flt.	in-flt filing	onbd aids	onbd par-proc	

On the topic of demonstrations, the panel was quite strong in its view that a successful, effective demonstration must reflect a real, not a toy, problem. Not only should a real-world problem be chosen, but real data should be used, and real interfaces supported. The principal constraint on the demonstrations should be one of bounding. The problem solution being demonstrated should be a very narrowly defined one, intended only to demonstrate the solution to the key issues under consideration.

The panel agreed with the objectives cited for the FAA's AERA-3 initiative, but expressed some concern over three issues. The time frame for the integration of AI technology into the ATC was not specified. The goals one would strive to achieve in this system will be very different if the realistic time frame is distant (say, 2020), rather than near (say, 1995). It may be desirable to have goal statements for multiple epochs. The panel felt some difficulty in dealing with a lack of specificity here. Second, substantial attention will have to be given to issues for functional integration up to the user (e.g., controller, flight crew) level. Finally, the ATC system is such a complex system that sophisticated multidisciplinary trade-offs must be conducted to optimize the system's performance around its functions and the technologies supporting them. It appears that this study is an element of such a series of trade-offs, and is accepted as such by the panel as an effective means of exploring alternatives for the future air traffic control system.

FINAL REPORT - GROUP 1

Ronald L. Larsen, University of Maryland

This report will be limited to the three major questions addressed by the group. While there was not unanimous agreement on the rank order of the priority areas, there was general agreement on the four major ATC functions that are potentially subject to substantial improvement through the application of AI technologies. These are: flight plan generation; flow control during the operational aspects of flight; real time conflict resolution; and the prediction and detection of severe weather conditions. These four areas are not only very important to the ATC system but could also benefit substantially from the application of artificial intelligence approaches.

The panel attempted to identify the critical problems of applying artificial intelligence successfully and beneficially. A very significant issue which was of great concern was verification and validation of AI software for flight rated requirements. This remains a challenging problem.

The interface to the human - the human on the flight deck or in the aircraft control station - was also identified as a major concern. How should the system communicate information to that individual? How can the individual interact with the system effectively? Analysis of the plans for AERA-3, for example, revealed a proposal for a nationwide system employing advanced technologies, such as cooperating expert systems with distributed knowledge bases. The design of such a system, with effective interfaces to its human operators, poses non-trivial problems requiring further research. Automated planning is a very complex problem involving geometrical and temporal reasoning. It also is a major AI technology for the future ATC system. For example, during a severe weather encounter, one must extract higher levels of information from the perceived or measured data, consider alternate courses of action, and quickly prepare response. Competitive goals interact when one is putting together a plan. In an environment where there are competing goals and limited resources, the planning problem becomes very difficult. Heuristics which are particularly suited to the operational environment of the aircraft control system are needed to converge quickly to satisfactory solutions. Another real issue that must be confronted during the development of some of these systems beyond the laboratory and into the engineering department is the flight-rating of this technology in order to get it into actual use.

To conduct effective technology demonstrations, the panel expressed a fairly strong feeling that there was a need to talk not about toy demonstrations but about real demonstrations similar to those discussed earlier. An effective demonstration must work with real data, work with real problems, and work with real interfaces. Care must be taken during problem selection to ensure the demonstration is feasible and achievable within the resource constraints of a technology demonstration.

DISCUSSION

Comment I would like you to expand on the area of flow control. We have been talking for years about the interfacing of AERA.

Ronald L. Larsen Dr. Campbell brought up this point in his presentation. The panel briefly discussed the AERA-3 objectives and supported them in principle. Would anyone on the panel choose to confirm or deny this? There was also discussion of the integration of traffic flow management. The integration issue of concern there is how does one marry those, a non-trivial problem for sure. There are many compelling technology problems which must be solved to realize the objectives of AERA-3. We are trying to identify some of them here.

Comment A specific time frame would facilitate decision making. Suppose, for example, we said, "What can we do over the next 20 years?" There may be other things that we will also want to look at beyond that time period that may satisfy some of the ATC concerns for making the system work better. Two time periods should have been examined, the next 10 or 20 years, and then beyond.

Comment We have a good forum looking beyond right now and beyond 2010.

Comment I understand today that the intent is not to control but to manage and hopefully reduce controls. In managing we expedite.

Comment In my view the set of functions that were listed were like separate little pieces of an air traffic controller. For an expert system to control some aircraft as a physical volume in space, operating within that paradigm, this particular set of names may not be the right set of names. There may be a need to discuss some more integrated controller functions. An individual controller performs both local flow control and conflict avoidance, and flight plan, tactical flight plan generation or flight path changes. My concern is the utility of breaking these apart into separate little pieces when perhaps the real concern should be with a system that performs all of the above.

Comment My concern has been that limitations of time may have forced the panel to try to reach conclusions quickly, maybe without having all the facts and an adequate discussion, and I am saying that in a positive sense. It was really a useful first step here but we should not conclude that we know the answers now. All this should do for us is motivate us to do something more and better.

Geoffrey D. Gosling Some critical AI technology has been identified but in order to implement it, there is other technology not considered as AI technology which is also critical to being able to implement it. For example, we do not really know as much as we ought to about system goals and measures of effectiveness. One of the things that you are going to have to deal with as we start looking at some of these AI techniques is that we can no longer ignore these by saying that we will continue doing it the way we have always done it. We must explicitly ask, "What is the trade-off between fuel economy and safety?" for example, if you are going to start writing algorithms that are making those trade-offs.

PRELIMINARY REPORT - GROUP 2

Alfred C. Robinson, Battelle Columbus Laboratories

Where and Why Are Expert Systems Applicable?

The group began with a consideration of what parts of the air traffic system seemed most promising as applications for expert systems (ES). After some discussion, it was concluded that all parts of the system were potential applications. A more fruitful point of view was a generic one of the types of problems (in each part of the air traffic control (ATC) system) which are amenable to ES treatment.

It was concluded that the following are candidates:

o Planning - Development of plans, especially short-range plans for any part of the system.

o Diagnostics/Maintenance - Use of ES in diagnostics has been proposed for many types of maintenance problems. This is a possible approach for all parts of the ATC system.

o Process Control - Control of all types of real-time processes, from the control of aircraft in the airspace to dispatching of maintenance vehicles and personnel are potential fields for ES.

o Training - Training of controllers, maintenance personnel, supervisory personnel and managers are all areas for application of ES. Not only training of new personnel, but re-training on new equipment and procedures could be considered.

In all these potential application areas, the reasons for using ES are somewhat similar. The principal ones identified were the following:

o Speed - Expert systems could deliver results more rapidly than human experts, given proper hardware and software design. Especially in control of aircraft, speed could be important.

o Capture of Expertise - Expertise is a perishable commodity. Experts move on to other responsibilities or into retirement, taking with them knowledge which properly belongs to the system. ES offers a means of capturing and sharing this expertise.

o Facilitation of Training - A closely related issue is upgrading of skills of new or less skilled personnel. ES can permit other personnel to learn from the best experts much more easily than personnel training, even when that training is carried out by those same experts. An expert is not necessarily a good teacher, but ES offers a means of combining the skills of good teachers with the knowledge of the best experts.

o Improved Understanding - In codifying expertise, much will be learned about the actual principles of operation of the ATC system, which is not now explicitly documented. Improvements in the system can be much better evaluated if the actual operation of the past and existing system is better understood.

Approach to Expert Systems Development

In developing ES, the general approach should be one of starting with a tractable problem and growing it. It was suggested that in-house people, with knowledge of the ATC system should be trained in ES development and utilized to implement the new systems.

Particular attention should be paid to selecting problems such that failure of the ES would leave the ATC system no worse off than it is now.

It is important to understand the domain of the ES and to suspend reliance on it, when domain boundaries are approached.

Research Needs

The research needs for ES applications to ATC problems are partly the same as the research needs for ES in general. The group identified the following as the principal areas for needed developments.

o Dynamic Data Base. A real-time ES airspace controller would need to operate from a data base. However, this data base could differ in important ways from those for other ES applications. The ATC data base would be updated continually and the updates would come from many sources. Some of these updates could result in emergency situations requiring responses from many parts of the system. Each item of update information would thus have to be examined to see if major revisions in system operation are required, or whether the updates could be accepted as part of routine operation. Erroneous data would be particularly troublesome in this regard.

Also, there is a problem in retention of data. The system might need to retain a certain amount of past data for reference in making current decisions. It would probably also be desirable to have some amount of permanent record storage for system evaluation/diagnosis and for use in accident litigation.

o Hardware Speed and Size. Real-time operation of a large expert system is on the fringes of the present state of the art. It is a simple matter to postulate systems which can not be supported by present capabilities. Advances in hardware power

and architecture will probably be needed before a full airspace controller system can be designed.

o Software Languages, Portability and Verification. The state of software may not be able to support a full ATC system. Improvements in all areas will be needed, but verification and validation will be particularly troublesome. Some support for the C language was expressed because of its high degree of portability.

o Distributed Expert Systems. One architecture concept to relieve some of the above problems is that of a distributed ES. Each individual ES would have a particular domain and the different systems would communicate with each other. Relatively little has been done in this area.

o Parallel Processing. This is a general tool for speeding up all types of computation-intensive activities. Distributed ES is one way of implementing this, but not the only one.

o Human Interface. There are many problems in interfacing an expert system with a human. In the ATC context, one of the more serious considerations is that of speed of communication. Another is that of making provision for human takeover of ES functions quickly and effectively. These aspects are not central issues in other applications of ES.

o Capture of Knowledge. This is a common problem in ES research, but in the ATC world there should be considerable emphasis on rapid updates of system expertise based on actual operational experience. The system could soon outgrow the experience base of the experts on whom the system is based.

FINAL REPORT - GROUP 2

Alfred C. Robinson, Battelle Columbus Laboratories

Group 2 began its work on what turned out to be a dead end. It attempted to list the elements of the air traffic control system (ATC) in which expert systems (ES) could make a contribution; it also listed the different functions which could apply to any one of these physical elements, such as operations, planning, maintenance and training. It was concluded very quickly that this was not a useful way to look at the problem, because all these functions in all the system components could potentially make use of artificial intelligence (AI).

A different and more fruitful approach was then adopted - that of trying to define the characteristics of problem areas in which AI could be applied effectively in the foreseeable future. To get at this, the reasons why one would look to an application of AI were examined. There were several of these.

o Capture of expertise. There are cases in which expertise is in short supply and it is desirable to make it more widely available. In particular, expertise is continually draining away because of personnel turnover.

o Augmentation of expertise. There is at least a prospect, based on experience with other systems, that an expert system could draw on many experts and produce a product better than any of the individuals contributing. Getting the same effect through a meeting of experts is extremely expensive and not feasible in many cases.

o Exceed human response times. Even in cases where the requisite expertise is available, there may not be time for humans to react. Expert systems might alleviate this limitation.

Based on these reasons the panel attempted to characterize places where AI was potentially applicable. There was agreement on the need to begin with a problem that seems to be tractable with present-day technology in expert systems and for which an expert system can be built in a relatively restricted period of time. It would be built, tested and improved. People would become comfortable with the approach and based on that confidence other related problems would be approached. Naturally, tasks which are primarily cognitive in nature would be sought.

Some non-trivial job would be sought, one which is normally performed by a person who is perceived as an expert. Also, cases would be sought in which a failure of the expert system would leave you no worse off than you were before.

The types of applications that met these requirements were planning, diagnostics and maintenance, process control and training. In the training field, primarily intelligent, computer-aided instruction was meant. It was concluded that there are probably hundreds of potential projects in these areas.

There was considerable discussion of approaches to problem selection and problem solution. There was some consensus favoring development of in-house capability in expert system development, rather than contracting for that capability with someone who is not familiar with the application area.

The shortcomings of current capabilities and research needs were then assessed. Seven areas requiring attention were identified:

o Dynamic data bases. One problem not widely addressed is that of data bases with rapidly changing content. This may be more of a problem for air traffic applications than for other types of expert systems. New data are continually entering and these data may be highly significant. It may be desirable to be able to recover the data base of three minutes ago or five minutes ago. Also, there is an issue of an expert system improving its solution over time, especially in the face of a changing data base. This type of problem needs more attention.

o Hardware speed and size. Obviously the bigger, faster and cheaper the hardware, the more problems can be addressed. AI shares these needs with other types of computation, but AI has, throughout its history, been limited by computer power. This is being worked by many organizations. There are some AI-specific machines, and these should be further developed for the AI community.

o Software. This was perhaps the most controversial item. Higher order languages, portability and verification and validation were the topics most frequently mentioned. Portability is needed for re-hosting of developed programs and some of the group saw this as the major problem in the software area.

o Distributed expert systems. The interaction of separate expert systems, with overlapping or non-overlapping domains seems to have application to the air traffic system.

o Parallel processing. This is one of the potential means for increasing the speed of the expert system.

o Human interface. The problem of how the expert system can deal with the human elements, especially in real-time situations, in which there is limited time for "discussion" is a severe problem in air traffic applications.

o Capture of knowledge. The techniques for capturing knowledge are still not well developed.

Much time was spent in developing a concise and well-reasoned list; this list seems to portray some agreement on what the AI community should be doing.

The group strongly supported the concept of starting small. There is frequently a tendency to ask for a largo, tightly spocified system which will be delivered in ten years and will do everything. This is quite contrary to the spirit and practice of development of expert systems where you start with something that works, but probably not very well. Then you play with it until you like it better.

The general precept of software engineering these days is to develop complete specification before the first line of code is written. This is not the way most software is developed. Expert systems methodology makes explicit the old cut-and-try approach by which most software for all purposes is developed. These days both approaches have strong and highly principled defenders.

Another point that should be noted is the interplay between research and applications. The panel felt that they must move together and that research should, at least in part, be motivated by what is needed for applications.

INDIVIDUAL INPUTS

Participants in the workshop were invited to submit written comments or papers on the issues discussed at the workshop for inclusion in the report of the workshop. The following represents the material received.

INDENTIFICATION AND DEVELOPMENT OF ARTIFICIAL INTELLIGENCE APPLICATIONS IN AIR TRAFFIC CONTROL AUTOMATION

Geoffrey D. Gosling, University of California, Berkeley

The significant increases in the level of automation of the United States air traffic control system currently being implemented or envisaged under the National Airspace System Plan suggest both a need and an opportunity to explore the potential for applying artificial intelligence (AI) technologies in the future ATC system.

Artificial intelligence is the name given to a broad field of computer techniques that have the general goal of developing "intelligent" processes which enable computers to perform tasks that usually require human skills, such as understanding language, pattern recognition, learning, problem solving, and so on. The field of artificial intelligence deals both with developing general methods for solving problems and with applications of these methods to specific domains of interest. Although recent interest has tended to focus on expert systems (computer programs that attempt to replicate the performance of human experts by means of rule-based reasoning), in part because of the availability of commercial software products, there is a wide range of other AI techniques and applications that may be relevant to ATC problems, including computer vision and speech recognition.

However, merely because certain computer techniques can be applied to ATC automation does not necessarily mean they should be. In the light of the complexity of the ATC system and the heavy costs of failures, in terms of human lives at risk and wasted resources, an assessment of the potential for introducing such techniques must answer two broad questions:

1) How can it be done?
2) What are the costs and benefits of doing it?

The first question must address not only how to make the techniques themselves work at the desired level of reliability and performance, but also how they can be fitted into the rest of the ATC system. The second question addresses the "value added" that can be obtained by using the techniques, compared to solving the problems in other ways.

The large number of potential AI applications in the ATC system that have been identified in research to date may be grouped into six categories:

- o Intelligent assistance
- o Strategic planning
- o Improved sensing and communication
- o Tactical control automation
- o Failure recovery management
- o System planning and training.

Intelligent assistance includes improved presentation of information, alerts for potential decisions and automatic execution of routine functions. Strategic planning techniques utilize knowledge representation and search methods to anticipate future conditions and regulate traffic flow to reduce aircraft conflicts and delay. Improved sensing and communications includes voice synthesis and recognition and computer vision. Tactical control automation applications utilize expert systems and heuristic planning techniques for conflict resolution, runway and airspace configuration management and dynamically adjusting the control rules. Failure recovery management includes both system monitoring and contingency planning techniques, as well as provision of real-time support to system managers attempting to redeploy resources. Applications in system planning and training include improved simulation tools and use of expert systems to assist in system configuration planning. The range of possible AI applications is summarized in Table 1.

In order to assess the usefulness of AI techniques in ATC, it is necessary to define the control environment within which these techniques might be applied. To illustrate the wide range of possible ATC environments, research at the University of California, Berkeley, has identified seven sample control strategies (1).

1) See and avoid, in which each aircraft is responsible for identifying and avoiding other aircraft through visual contact;
2) Collision avoidance, in which on-board systems monitor the position of nearby aircraft electronically and provide flight crew guidance for evasive action;
3) U.S. Today, in which ground based

Table 1. Potential Applications for Artificial Intelligence in Air Traffic Control.

Area	Application
1. Intelligent Assistance	o Improved Presentations of Information o Alerts for Potential Decisions o Menus of Alternative Actions with Recommendations o Automatic Execution of Routine Functions
2. Strategic Planning	o Extended and Coordinated Probes o Deconflicted 4D Flight Plan Generation o Improved Interface with Tactical Control o Fuel Analysis as Part of Control Decisions o Demand Responsive Scheduling o Aircraft Specific Delay Allocation o Airport Capacity Forecasts
3. Improved Sensing and Communication	o Voice Synthesis and Recognition o Automatic Clearance Transfer o Computer Vision
4. Tactical Control Automation	o Collision Avoidance Direction o Runway and Airspace Configuration Management o Automated Decision Making o Flexible Control Rules
5. Failure Recovery Management	o System Monitoring and Crisis Anticipation o Contingency Planning o Failure Recovery Support and System Configuration Selection o System Restoration o Major Disruption Response
6. System Planning and Training	o System Configuration Planning o Improved Simulation Techniques o Pseudo-pilot Automation

controllers monitor aircraft with radar, supported by partially automated data processing, and issue clearances by radio;

4) AERA Stage 1, in which the current system is supplemented by improved communications and controller support functions;
5) AERA Stage 2, in which the computer would detect and resolve aircraft conflict, automatically generating the appropriate clearances;
6) Deterministic, in which advanced aircraft flight management systems would permit aircraft to follow approved deconflicted four-dimensional flight paths, with ATC intervention only to handle unplanned deviations;
7) Integrated, in which deconflicted four-dimensional flight plans are adjusted on a real-time basis to respond to changing conditions.

Although conventional approaches are capable of making the first four strategies function, it appears that AI techniques may enable higher performance to be achieved. In contrast, it appears that the last three strategies may require application of AI techniques to enable their implementation.

Evaluation Methodology

In view of the large number of potential applications of AI techniques and the high cost of developing and testing prototype software, there is a critical need to develop improved methods for performing preliminary evaluations of proposed ATC system enhancements. Most conventional ATC simulation software has the disadvantage of being designed to replicate the performance of the existing system and being very costly to modify.

An evaluation methodology has been proposed for assessing the potential application of particular AI techniques, based on a computer model that can be easily configured to simulate the functioning of any specified ATC system and record the necessary data to assess the performance of the system (2). The proposed model operates by simulating the behavior of generally defined entities, termed actors, that represent components of the ATC system, such as controllers, aircraft, and computers. Each actor behaves according to specified behavioral algorithms that select from a set of permissible actions in order to meet defined objectives. The actors exist within an environment which has spatially and temporally definable properities, such as wind or precipitation, which affect the outcome of any given action by an actor. In addition to simulating the operation of the system, the model also records the data necessary to assess the system performance. Record keeping functions monitor the flow of requests, instructions, and information and record them in a data base for subsequent analysis.

Model runs are performed by specifying the characteristics and objectives of each actor, as well as the rules that govern the behavior of the simulation. The physical and other conditions of the environment must also be specified. High (strategic) level control of the simulation is achieved by specifying a detailed set of objectives for certain classes of actors, as represented, for example, by aircraft flight plans, but the moment to moment conduct of events is entirely in the hands of the behavioral and performance algorithms.

Because of the potentially catastrophic consequences of a failure of the ATC system, evaluation of particular techniques should address how the system will handle and recover from failures. Conventional failure analysis has severe limitations for use in the type of evaluation proposed, and alternative approaches need to be developed.

In order to evaluate any proposed system, a set of measures of effectiveness must be defined that reflect the objectives of the air traffic control system. These measures must address issues of

- safety
- efficiency
- economy
- equity.

Safety measures include collision risk, proximity measures (number of near mid-air collisions and separation violations), and measures of pilot and controller workload. Efficiency measures describe how well the available airspace is utilized for the movement of aircraft, and can be expressed in terms of the number of aircraft-miles flown per hour per unit volume of airspace. Economy measures reflect the resources required to operate the system and include both user costs, such as fuel consumption and travel time, as well as ATC system equipment and operating costs, together with the schedule disruption impacts of delays imposed on the system. Equity considerations address how the costs and benefits of the system are distributed among the various classes of user, taking account of both the type of user (air carrier, general aviation, and military) and the size of aircraft.

Conclusions

The current state of the art of artificial intelligence techniques appears to support a wide range of possible applications to air traffic control under different control strategies. It appears that techniques developed for heuristic search and the creation of expert systems offer the most promise for near-term application. Areas such as speech interpretation and computer vision may offer potential applications further into the future.

Alternative control strategies can be identified that offer a wide range of both performance and implementation cost, suggesting that different strategies would be appropriate under different circumstances. The development path for future ATC systems should therefore address what mix of strategies is most appropriate under the expected circumstances. Artificial intelligence techniques appear to have potential application in all types of control strategy, including those with less automation than envisaged under the National Airspace System Plan. In addition to applications providing support to controllers for real-time control of air traffic, such as intelligent displays and monitoring/planning functions, there appear to be useful applications for expert systems in areas that include failure recovery management, traffic flow control, training, and airspace configuration planning.

It should be recognized that to achieve the full benefits of AI techniques, it may be necessary to change significantly either procedural rules or the way ATC services are provided. In order to assess the cost and benefits of such techniques and changes, there is a need for significantly more sophisticated evaluation tools than exist at present, with particular attention being paid tc analyzing complex failure modes involving equipment, software and human operators, and developing clearly defined and accepted measures of system performance. Critical issues remaining to be addressed include

- the appropriate role for the human controller in a more automated control system;
- software testing and certification procedures;
- formal representation of how controllers and other personnel actually do their existing jobs;
- specification of requirements for developing the capabilities to implement AI techniques in ATC system software;
- resolution of trade-offs between conflicting objectives of the ATC system.

In the light of the massive investment being made in the ATC system modernization, it would appear highly appropriate to devote considerable resources as a matter of some urgency to a long-range research program directed at the problems to be faced in the future evolution of the ATC system.

It should be recognized that the skills necessary to perform research of this type are in short supply, given the technical complexity of the ATC system and the intensely competitive market for those with experience developing AI applications. If FAA and its contractors are to be able to draw on a diverse and well-founded body of knowledge and scientific personnel with the skills to support long-term development of the ATC system, it is important that current basic research efforts are expanded and sustained. In particular, adequate funding should be directed at universities in order to ensure a stable supply of Ph.D. students with the necessary background and skills. Although not requiring large amounts of money in comparison to other research and development efforts, these programs must be sustained at realistic levels over the long-term in order to be effective.

References

1. G. D. Gosling and S. L. M. Hockaday, Identification and Assessment of Artificial Intelligence Techniques in Air Traffic Control, Research Report UCB-ITS-RR-84-14, Institute of Transportation Studies, University of California, Berkeley, September 1984.

2. G. D. Gosling, Development of a Computer Evaluation Model to Analyze Alternative Air Traffic Control Strategies, Working Paper UCB-ITS-WP-84-10, Institute of Transportation Studies, University of California, Berkeley, November 1984.

THE USE OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN THE FEDERAL AVIATION ADMINISTRATION SYSTEMS

David A. Spencer, Massachusetts Institute of Technology, Lincoln Laboratory

Software techniques and development methodologies from artificial intelligence have a role to play in the National Airspace System as this system becomes more automated. However, the applications must be carefully chosen. Artificial intelligence (AI) techniques must always be compared with more conventional "algorithmic" methods for achieving the required behavior. Where an algorithmic approach is known, it is generally preferable to an AI approach. At the opposite end of the spectrum of complexity, there are behaviors which may be desirable but which cannot be achieved with any current technology. Both the "algorithmic" boundary and the "impossible" boundary are fuzzy and change over time. Research is often needed to determine where a specific task lies. AI techniques appear to have advantages in areas where the necessary flow of control is very complex, or not well understood, or where the system is likely to be extensively modified during its lifetime in unforeseen ways, or where it is desirable that the reasons behind the system's behavior be understandable to users. The latter might include giving the system the ability to answer questions about why it took a certain action.

Software Development Environments and Methodologies

The software development environments and methodologies used by AI researchers are important in their own right, whether or not the system being developed uses AI techniques. These include powerful personal work stations, such as LISP machines and the highly integrated environments created by the LISP machine software systems. Awareness of and experience with these capabilities should be encouraged within the Federal Aviation Administration (FAA). Although the FAA normally does not develop its own software, it would benefit from use of similar techniques by its contractors.

One interesting software development methodology is that used in creating expert systems. This involves quickly creating a prototype design, using some representation such as production rules that is modified. This prototype is then tried out on test problems and the results criticized by an expert. If errors are found it is relatively easy (compared with conventional implementation in a procedural language) to find the offending rule or fact and modify it to get the desired behavior. This approach has two advantages. First, it exposes the design to criticism by the eventual users or other knowledgeable persons early in the design phase. Second, this criticism is based on actual performance of the prototype on test problems, rather than on a system description in a specification or design review document. Over time, these test problems are made harder and more realistic, ultimately becoming the actual production environment.

Functional Specifications

While such a prototype may not satisfy all the system requirements in terms of real time performance, capacity, or resource usage, it does provide the basis for a functional specification of the system that is known to work as desired. This functional behavior can then be reimplemented, if necessary, to achieve performance goals. This approach explicitly recognizes that a certain amount of debugging of the requirements, the specification and the design always occurs, and that it is better to do this in a low overhead situation rather than at the point where dozens of contract personnel are trying to implement the operational version of the system.

This approach can be contrasted with two alternatives. In the first the requirements are very broadly stated, the contractor is given considerable latitude in how to meet those requirements, and it is not always clear whether or not a particular approach will ultimately satisfy the requirements. A rapid prototype effort would resolve the requirements in more detail and settle critical aspects of the technical approach before the detailed design is begun.

The second alternative is that a large, detailed specification is written by one contractor and another is hired to implement it. The specification may not be validated against realistic scenarios until the production system enters acceptance testing.

Rapid prototyping such as is proposed here is particularly efficient when done in an environment such as a LISP machine work station, and is facilitated by techniques developed by AI researchers for representing data and procedures in an easily modified manner. This combination of capabilities provides a software environment where rapid implementation and modification of a prototype are feasible, and provides a flexible inteactive graphics interface which can be programmed to simulate functionally any necessary displays and controls. The gain in flexibility is paid for by greater resource usage, by lower system capacity or slower response time, and by lack of mechanical (as opposed to functional) realism in the man-machine interface.

In summary, there are beneficial side effects from AI research in the area of software engineering. These are independent of whether or not AI techniques are used in implementing the operational software.

ARTIFICIAL INTELLIGENCE AND AIRSPACE MANAGEMENT

Paul C. Leonard, Air Transport Association of America

Artificial intelligence (AI) introduction into airspace management must not attempt to automate the controller. Just as the flight crew's functions have changed with the introduction of more sophisticated flight management systems in current and future aircraft, airspace managers will most certainly function differently than controllers do today. The flight management systems on aircraft will need to interface with the ground airspace management system with pilots and airspace managers making virtually none of the routine decisions.

Given that airport capacity problems will be addressed by more efficient use of reliever airports, improvements in the utilization of existing airports, and development of some new airports, unconstrained operation should be the rule rather than the exception. Current Federal Aviation Administration (FAA) planning documents such as the National Airspace System Plan and the Research, Engineering and Development Plan do not presently reflect this notion. Rather, they reflect a design philosophy of continued constraint to the airspace users. This

notion is particularly reflected in the Research, Engineering and Development Plan which is driving toward an ultimate system called "Flow Management". The message which must be stated emphatically is the need to influence the direction which system planners within and outside the FAA must take to incorporate AI efficiently into airspace management. Again, a goal of unconstrained operation to airspace users must be the rule.

ARTIFICIAL INTELLIGENCE AND OTHER ASPECTS OF AIR TRAFFIC CONTROL

Robert W. Crosby, Federal Aviation Administration

Much of the attention of the artificial intelligence (AI) workshop was focused on direct air traffic control (ATC) issues: conflict resolution, flow control, and weather prediction. This was entirely proper, and the Federal Aviation Administration (FAA) advanced automation program fully concurs with this emphasis. However, rather than reiterate the contributions of others, it would be preferable to use this opportunity as a means of establishing the potential benefit of AI in some of the less direct, but equally important, aspects of ATC. Specifically, the following three topics are suggested: software (SW) design, system repair and maintenance, and training.

Software Design

The reliability of the advanced automated system (and its successors) will be critically dependent on the SW design. Because of the extremely high reliability desired, on the one hand, and the complexity of the SW on the other, it is essential that techniques be used that: 1) minimize the number of hidden faults inadvertently designed into the system; and 2) provide fault tolerance for those that remain. Existing design methods may be enhanced substantially by incorporating AI techniques. Two such techniques come to mind immediately: the use of intelligent search techniques to explore a branching SW tree; and knowledge based systems that make use of expert techniques to solve complex SW design problems.

System Repair and Maintenance

The availability of the ATC system depends critically on rapid failure detection, isolation, and repair. As experience is accumulated on failures, it is likely that this knowledge can be incorporated into an expert system that will reduce system repair time significantly. A second area relates to the detection of incipient expert failures. Again, based upon accumulated knowledge, it should be possible to anticipate many hardware (HW) failures with aid of an expert system. As an aid to maintenance personnel, AI techniques can improve both system performance and personnel productivity.

Training

For the foreseeable future the ATC system will be operated primarily by controllers, with automation being used to aid them, particularly in performing routine tasks. The training of controllers, as well as the operating and maintenance personnel who support them, is thus a key link in the performance of the system. Computer based instruction (CBI) has been used by the FAA for over a decade in the training of these personnel. However, existing CBI, through rote learning techniques, seeks primarily to reduce the number of instructors required for training. Although rote learning may be suitable for routine tasks, the successful operation of the ATC system also requires, from time to time, innovative solutions to new or unpredictable events. As the degree of automation of the ATC system increases this need can be expected to increase. CBI based upon rote learning tends to reject those people who are good at innovation, but bored by routine. Obviously, the ATC system needs both types of people. New and more powerful CBI techniques are now being explored that make use of AI techniques to provide a more versatile learning environment. The development of such a training system for the FAA should be given high priority.

AIRPAC: ADVISOR FOR INTELLIGENT RESOLUTION OF PREDICTED AIRCRAFT CONFLICTS

Curtis A. Shively, Mitre Corporation

SUMMARY

AIRPAC is an expert system being developed to assist air traffic controllers with the planning of resolutions for predicted violations of safe separation or "conflicts" between aircraft. AIRPAC uses knowledge-based system (KBS) techniques to suggest aircraft maneuvers that will prevent a conflict. AIRPAC's choice of a resolution is based on decision rules gathered via consultations with air traffic controllers. By applying these rules to a description of the conflict, AIRPAC produces a single "best" resolution that includes detailed parameters of the recommended aircraft maneuvers. To plan resolutions, AIRPAC uses a hierarchical approach similar to the nested levels of abstraction in a human reasoning process. AIRPAC explains its operation by providing an audit trail of rules used in the search for a resolution. This explanation capability and the representation of resolution rationale in symbolic terms natural to humans are significant benefits provided by the KBS approach.

Introduction

The Federal Aviation Administration (FAA) is undertaking the development of the automated en route air traffic control (AERA) system (1). AERA is intended to automate many of the routine tasks performed by today's air traffic controllers. AERA will also provide computer-based tools for assisting controllers with the more complex planning and control functions requiring human intervention. An important purpose of the U.S. air traffic control (ATC) system is to assure that aircraft are safely separated from one another. An objective of AERA is to predict potential violations of safe separation or "conflicts" between aircraft ten to twenty minutes in advance. These predictions will be based on aircraft flight plans, wind observations, anticipated pilot or controller actions, and other information. If a conflict is predicted with sufficient lead time, AERA can suggest aircraft maneuvers to resolve it in a way that reflects desirable considerations beyond avoidance of imminent collision. The capability to predict aircraft conflicts and plan their resolutions in advance is expected to increase controller productivity and permit more airspace users to fly the routes they prefer.

Some previous work has been done toward automating the resolution of aircraft conflicts. FAA sponsored research on developing the AERA system has included the experimental implementation of a conventional numerical algorithm for selecting conflict resolutions based on numerical weighting factors. Some research on using KBS techniques for planning conflict resolutions has been done in the university environment (2) (3). The AIRPAC system (4) (5) described in this paper is the result of an independent research and development project conducted by the Mitre Corporation to investigate the general feasibility of applying KBS techniques to automation of ATC functions. AIRPAC includes knowledge gleaned from previous work on automating conflict resolution, as well as from new consultations with air traffic controllers who were shown the program at various stages of its development.

AIRPAC's Approach

AIRPAC assumes that other processes can model aircraft trajectories and predict future aircraft conflicts. AIRPAC focuses on the selection of aircraft maneuvers that will resolve the given problem. Some simple approximation of resolution trajectories is done by AIRPAC in order to check basic feasibility of a proposed resolution. However, it is assumed that AIRPAC's resolutions are passed through other trajectory modeling and conflict prediction processes to be certain that the given problem would be resolved and no new conflicts would be generated.

As of this writing, AIRPAC deals with the following types of conflict situations:

- one-vs-one - a single conflict between two aircraft, isolated from their other conflicts in the ATC sector
- one-vs-two - two conflicts, separated somewhat in time and space, but sharing a common aircraft
- three-at-once - three conflicts among three aircraft, each in conflict with the other two at about the same time and place.

A resolution consists of maneuvers for one or more aircraft that will prevent the predicted violation(s) of safe separation. In the en route phase of flight, two aircraft are considered to be safely separated if they are five nautical miles apart in the horizontal dimension or 1,000 feet (2,000 feet at high altitudes) apart in the vertical dimension. A conflict occurs between the two aircraft if separation criteria are violated in both dimensions at the same time. Therefore, resolution techniques are intended to assure separation in one or the other of these two dimensions.

Resolution Tactics

AIRPAC uses the following basic resolution tactics:

Horizontal
- Delay vector - turn off route, parallel, then back to route
- Delay route bend - continue original heading, delaying turn
- Vector around - turn off route, pass parallel to conflicting aircraft then back to route
- Vector both around - both aircraft, to opposite sides of route
- Vector behind - turn toward, then behind conflicting aircraft
- Vector cutting bend - turn and proceed directly to the navigation fix after a route bend.

Vertical
- Change altitude - climb (descend) to new higher (lower) cruise altitude
- Restrict climb - level off to pass below conflicting aircraft
- Restrict descent - level off to pass above conflicting aircraft
- Early descent - start descent early, to pass below conflicting aircraft.

AIRPAC can also recommend a speed change to achieve a resolution by merely altering the timing of an aircraft along its original path.

Each basic tactic shown above may imply a sequence of several maneuvers. For example, the restrict climb tactic includes a maneuver to level off followed by a maneuver to resume climbing once the conflicting aircraft has passed safely above. AIRPAC is intended to plan resolutions well in advance, before the conflicts represent imminent problems. Consequently, AIRPAC suggests a complete sequence of maneuvers to avoid the conflict and return any diverted aircraft to its originally intended route, destination, altitude or altitude transition.

Resolution Factors

AIRPAC attempts to assess the relative merits of various alternative resolutions. Each alternative represents choosing both the aircraft (one or more) to divert and the corresponding tactic to use. In AIRPAC these choices are based on factors such as the following:

- Conflict geometry - crossing, head-on, merging, overtake
- Aircraft intent - arrival, departure, overflight
- Aircraft speeds and relative positions
- Aircraft performance characteristics and limits
- Bends in aircraft route
- Intent and destination after the conflict
- Lead time before the conflict.

In considering these factors, it is desirable to recognize particular aspects of the conflict situation that might immediately suggest a good resolution. Suppose, for example, that an aircraft has a bend in its route shortly after the conflict region. An experienced human controller might immediately consider the possibility of diverting that aircraft directly to the next navigation fix after the route bend. Such a tactic would provide the double benefit of both preventing the conflict and shortening the path of the diverted aircraft to its desired destination. In this manner, the human controller has formulated a plan that considers broad, non-numeric aspects of the conflict situation and achieves multiple goals.

AIRPAC attempts to embody such reasoning abilities exhibited by human controllers. To that end AIRPAC's approach to finding resolution is based on the following principles:

- Recognize a multi-conflict situation (three-at-once or one-vs-two) and try to resolve it as a complete set.

- If this fails, try to resolve the individual one-vs-one conflicts in their order of occurrence in time.
- Whether a set is multi-conflict or one-vs-one, look first for a particular aspect of the situation that suggests a tactic achieving several goals at once.
- At each decision point, consider first the alternative which is believed a priori to be best.
- Stop the search as soon as a feasible resolution is found.

The decision to stop AIRPAC's search with the first successful resolution is somewhat arbitrary. Whether the controller should be presented only the "best resolution" or several viable alternatives remains an open question.

Hierarchical Planning

In AIRPAC the planning of resolutions proceeds according to the following steps:

- Problem decomposition
- Tactic selection
- Tactic development
- Maneuver parameter calculation.

These steps represent a hierarchy or different levels of abstraction, corresponding to refining and specifying the resolution in more and more detail.

In the problem decomposition step AIRPAC recognizes whether the given conflict set involves multiple conflicts. For a three-at-once conflict set, AIRPAC suggests a strategy for reducing the problem to a single one-vs-one conflict by maneuvering one particular aircraft away from the situation. In the case of a one-vs-one conflict set, the common aircraft is initially designated as the aircraft to be maneuvered before tactic selection begins.

During tactic selection, AIRPAC evaluates various alternatives for choice of aircraft to maneuver (unless already specified) and tactic to use. Alternatives initially considered are motivated by special factors such as aircraft route bends, if present. Otherwise tactics are proposed according to various cases of conflict situations, characterized by combinations of conflict geometry and intent of the involved aircraft.

Each resolution alternative selected for consideration is further evaluated by the tactic development process. In this phase a basic tactic such as "change altitude" may be further expanded into two possible subtactics, "go higher" and "go lower". Checks on the feasibility of the tactic are performed. For example, in the case of a go higher tactic, the intended altitude would be checked against the maximum cruise altitude of the given type of aircraft. Such checks may be done both before and after maneuver parameter calculation.

During maneuver parameter calculation parameters for the individual maneuvers in the resolution tactic are determined. These parameters include maneuver start and end times, turn angles for horizontal vectors and restrictions of target altitudes for vertical maneuvers.

Knowledge Representation and Application

Much of AIRPAC's resolution knowledge for problem decomposition, tactic selection and tactic development is represented by IF-THEN production rules. If all antecedent clauses of a rule are satisfied, THEN the rule can be "fired" and its consequent clause(s) carried out. As of this writing AIRPAC includes 98 such rules.

AIRPAC's rules operate on information stored in data structures known as frames (6). In a frame the individual piece of information about a particular entity is stored as values in "slots" accessed via reference to the frame name. AIRPAC uses frames to represent the description of the given conflict problem and the details of the resolution being planned. Static knowledge such as the performance characteristics of various types of aircraft are also stored in frames. AIRPAC's rules themselves are actually represented in frames also.

An example of an AIRPAC rule (slightly simplified) is shown below:

```
IF (in-frame CONFLICT involved -ac -a)
AND (in-frame -a route-bend-after-conflict)
THEN (try vector-cutting-bend with air-
   craft-to-maneuver -a).
```

This rule applies if an aircraft involved in the conflict has a route bend after the conflict. If such an aircraft is found, the rule consequent ("THEN" clause) recommends resolving the conflict via a "vector cutting bend" tactic supplied to that aircraft.

The operation of this rule is as follows. The first antecedent of the rule examines the frame for the conflict to be resolved, shown here as frame "CONFLICT" for simplicity. In that frame are stored the names of the two aircraft involved in the conflict. The variable "-a" is associated with one of these aircraft and its frame examined by the second antecedent. If that aircraft has a route bend after the conflict, the rule can be fired and its consequence carried out. Otherwise the test is repeated for the other aircraft in the conflict.

For convenience in focusing the scope of rule applications AIRPAC's rules are grouped into rule sets. At each step of resolution planning, AIRPAC considers rules in only a single rule set for possible firing. Since all rules in a set are not necessarily mutually exclusive, more than one rule could be eligible for firing (all antecedents are satisfied). AIRPAC prefers to fire rules in the order in which they are originally defined to be members of the rule set. Thus the relative preference desired for various resolution alternatives can be represented by the ordering of rules within AIRPAC's rule sets.

Many of AIRPAC's rules are designed to explicitly redirect the search for rules that can be fired. For example, the "try" term in the consequent of the above rule transfers the search to the "vector-cutting-bend" set of rules. The "with" clause in that rule's consequent designates the aircraft with the route bend as the "aircraft-to-maneuver" via the vector cutting bend tactic.

As of this writing AIRPAC includes 26 rule sets which are organized according to the hierarchical planning steps outlined above. AIRPAC's control mechanism begins the search for a resolution in the set of problem decomposition rules. These rules recognize whether the basic conflict situation is three-at-once, one-vs-two, or one-vs-one and direct the search to the corresponding rule set for tactic selection. A single tactic applied to a particular aircraft may be immediately proposed for evaluation, if that aircraft has a special attribute such as a route bend. Otherwise tactic selection is directed to a rule set designed for the given conflict geometry, i.e., crossing, head-on, merging or overtake.

Rules in the set for a particular conflict geometry are mutually exclusive, corresponding to different combinations of aircraft intent (e.g.,

arrival versus departure) and other factors. The rule that applies to the given situation suggests a list of resolution alternatives (both tactic and aircraft) in the order preferred for evaluation.

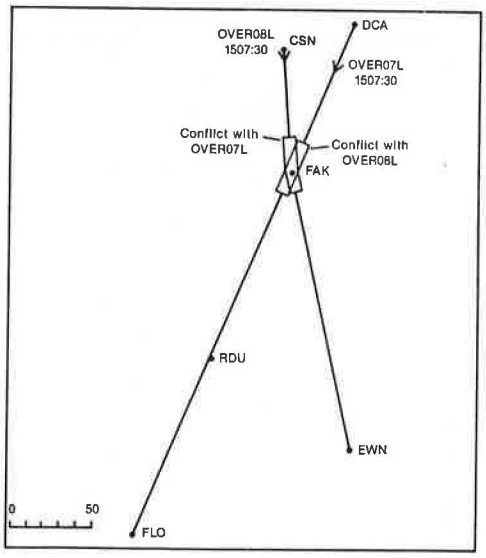
Evaluation of a resolution alternative is accomplished by rules in the development rule set for the proposed tactic. If the tactic passes all tests and parameters for its maneuvers are computed successfully, the alternative becomes the recommended resolution and the search is terminated. If necessary, each proposed alternative is evaluated in turn, until a successful one is found or all are exhausted. If this list is exhausted without success, AIRPAC terminates the search without finding a resolution to the conflict. In this manner AIRPAC employs a depth first method to search forward from the description of the conflict to details of a resolution plan.

Implementation

AIRPAC has been implemented using an integrated representation and inferencing system (IRIS) (7) developed at Mitre. Written in LISP, IRIS provides a variety of programming paradigms based on a common underlying frame representation language (FRL) (8). In addition to production rules and forward chaining IRIS also supports objects, active values and of course the usual procedural programming capabilities of LISP. AIRPAC was originally programmed in Franz Lisp (9) on a VAX 11/780 computer in Mitre's command and management information systems (CAMIS) laboratory. The software for AIRPAC has also been translated into Zetalisp (10) for execution on a special purpose LISP computer.

At no time during the development of AIRPAC have any attempts been made to optimize its run time efficiency other than executing the LISP code compiled rather than interpreted. However, it is interesting to note that for the example given later in this paper execution time on the VAX 11/780 was about 30 seconds. This was reduced to just several seconds on the LISP computer. Further execution speed improvements are likely to result from eliminating some features of IRIS, included for generality but not used by AIRPAC.

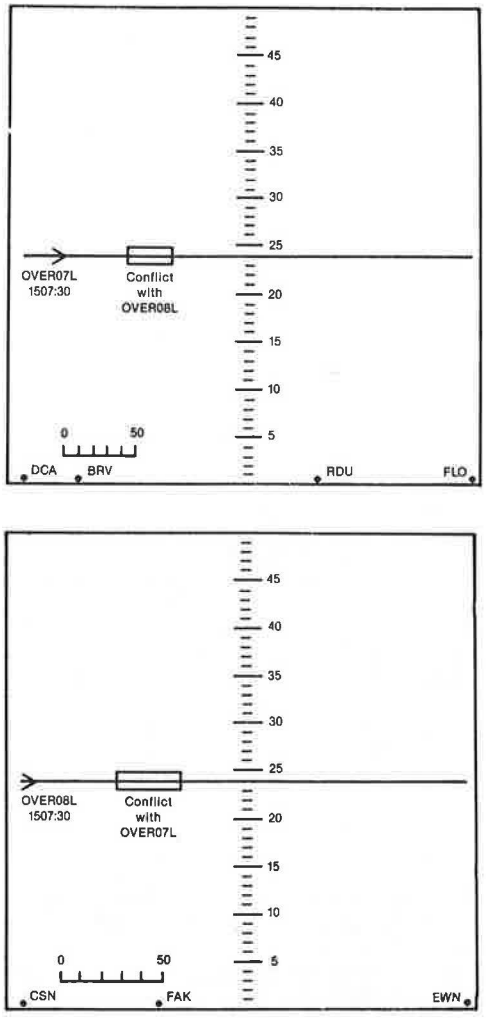
Figure 1. Horizontal plan view of aircraft trajectories in a conflict scenario.



OVER07L
DCA BRV RDU FLO 24000 240
OVER08L
CSN FAK EWN 24000 240

The Facts:
Conflict-Type Crossing OVER07L OVER08L
Intent Overflight OVER07L
Intent Overflight OVER08L
Time-to-Conflict 10.0
Start-of-Conflict 15 17 29
End-of-Conflict 15 24 50
Time-at-Crossing OVER07L 15 21 25
Time-at-Crossing OVER08L 15 21 44

Figure 2. Vertical profile view of a conflict scenario.



Examples of Conflict Resolution

An example of a one-versus-one conflict situation is shown in Figures 1 and 2. Figure 1 shows a horizontal plan view of the aircraft trajectories and Figure 2 shows altitude versus distance along route for each aircraft. On the horizontal view a "conflict box" encloses that part of each aircraft's route wherein a violation of safe horizontal separation from the other aircraft is predicted to occur. On the profile view of each aircraft the conflict box illustrates a vertical protection buffer around the altitude of the other aircraft while the two will be violating the horizontal separation standard. Arrows in both figures show the position of each aircraft at the time (15:07:30) when the resolution planning is being done.

Figure 3. Example of AIRPAC solution tree.

```
0.  solution
    1.  solve-conflict-set                          success
        2.  one-versus-one                          success
            3.  ovo-crossing                        success
                4.  vector-behind OVER08L           success
                4.  delay-vector OVER08L            success
                    5.  right OVER08L               success
                    5.  parallel OVER08L            success
                    5.  left OVER08L                success
```

Salient facts describing the conflict situation are shown at the bottom of Figure 1. Two overflight aircraft, OVERO7L and OVERO8L, are involved in a crossing conflict where their routes interact near the navigation fix at FAK. Route information indicates that both aircraft are level at 24,000 feet and traveling at 240 knots. The starting time of the conflict is indicated as 15:17:30, giving a lead time of about 10 minutes to plan and execute a resolution. Note from the "time-at-crossing" facts that OVERO8L is predicted to reach the route intersection point about 19 seconds later than OVERO71,

Figure 3 shows AIRPAC's resolution to this conflict in nested levels corresponding to the steps in hierarchical planning. From the figure it can be seen that the search progressed through rules in the "solve-conflict-set", "one-versus-one" and "ovo-crossing" rule sets. AIRPAC first evaluated a vector behind tactic whereby aircraft OVERO8L would turn to its left just prior to the conflict region and pass behind OVERO71. This tactic was declared to be a failure.

However, the second alternative, a delay vector for OVERO8L, passed all tests and thus the search terminated with success at all levels of the planning. A complete trace of the rules triggered (omitted here for brevity) would reveal that AIRPAC gave preference to maneuvers for OVERO8L because it was predicted to be later at the route intersection than aircraft OVERO7L.

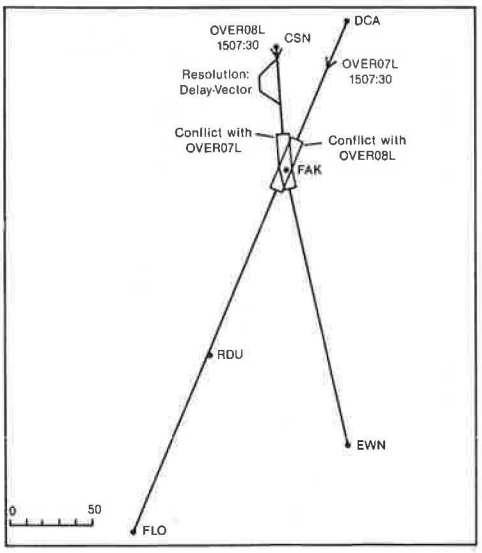
The delay vector tactic includes a turn right, turn parallel, followed by a turn left to rejoin the original route. Figure 4 shows a horizontal plan view of the conflict situation with these resolution maneuvers included. Parameters for the resolution have been selected so that aircraft OVERO81 will reach the route intersection at the revised time of 15:23:55, 2.5 minutes later than OVERO71. At aircraft speeds of 240 knots, this amount of delay corresponds to a distance of 10.0 nautical miles.

Several additional features of AIRPAC's user interface are illustrated in Figure 5. The "form-clearance" command displays maneuver parameters in terminology similar to that found in ATC clearances. The "why not" command may be used to obtain clarification for the reason AIRPAC rejected a resolution alternative. For the conflict just illustrated this command displays the "because" command attached to the failure by the inferencing process and the name of the specific rule, "vb-too-shallow" that caused the rejection. The user could then give the "print-rule" command to examine the form of this rule. After studying the rule and realizing its effect for crossing conflicts, the user may decide that a threshold value other than 90.0 degrees may be desirable. A rule editor has been implemented to permit altering the form of AIRPAC rules without leaving the LISP environment.

Observations

Although the development of AIRPAC is not complete, the following potential benefits of using

Figure 4. Horizontal plan view of conflict situation with AIRPAC's resolution trajectory.



OVER07L
DCA BRV RDU FLO 24000 240
OVER08L
CSN FAK EWN 24000 240

The Facts:
Conflict-Type Crossing OVER07L OVER08L
Intent Overflight OVER07L
Intent Overflight OVER08L
Time-to-Conflict 10.0
Start-of-Conflict 15 17 29
End-of-Conflict 15 24 50
Time-at-Crossing OVER07L 15 21 25
Time-at-Crossing OVER08L 15 21 44

Figure 5. Examples of AIRPAC explanation Commands.

```
1.   (form-clearance)

For aircraft OVER08L:

  ( (at (15 7 30) turn right to heading 220
      for 3 minutes)

    (parallel original route heading 175
      for 1 minute)

    (at (15 12 6) turn left to heading 130
      for 3 minutes) )

2.  (why-not 'vector-behind')

vector-behind failed because -31 is too
   shallow an angle to vector behind.
This failure was caused by rule vb-too-
   shallow.

3.  (print-rule 'vb-too-shallow)

(if (encounter-angle -a)
    (not (greater (abs -a) 90.0) )
       then
    (failure because -a is too shallow an
       angle to vector behind) )
```

KBS techniques for automation of aircraft conflict resolution have already been observed:

- KBS methods allow decision rationale to be expressed in symbolic rules rather than being limited to purely numeric expressions. Consequently, AIRPAC's rules relate the selection of resolutions to factors like type-of-conflict (crossing, head-on, merging, overtake) that humans can readily understand.

- User-readable symbolic rules have facilitated the gathering of expert knowledge about conflict resolution. Air traffic controllers not familiar with KBS technology have been able to suggest changes to AIRPAC's decision rationale and confirm the desired results in an on-line computer laboratory environment.

- KBS methods inherently provide a separation of conflict resolution domain knowledge from the inference mechanism used to apply the rules to a given conflict. Thus, it would be possible to tailor resolution strategy to particular ATC jurisdictions after field introduction of the AERA system, by minor changes to the rules, rather than extensive changes to the software.

- AIRPAC supplies a trace of what rules were triggered to give the result for a particular conflict. Such an explanation capability may help air traffic controllers to understand and accept recommendations provided by an automated conflict resolution function.

Further Work

As of this writing AIRPAC can select resolutions and compute details of maneuver parameters for a variety of conflicts involving only two aircraft. AIRPAC also handles some multi-aircraft, multi-conflict situations by decomposing them into similar conflicts to which the two-aircraft techniques can be extended. AIRPAC needs to include more methods for providing a comprehensive resolution to a multi-conflict situation treated as a single problem. Where this approach fails, AIRPAC must coordinate the search for resolutions to conflicts that are related. This coordination might be accomplished by global consideration of constraints that resolutions to the individual conflicts impose on each other.

Other factors represent more localized constraints on the resolution maneuvers of individual aircraft. AIRPAC does presently consider the limitations of aircraft performance characteristics. However, AIRPAC should be expanded to deal with constraints imposed by winds, severe weather, aircraft traffic flows, ATC sector boundaries and procedural restrictions.

In the ATC operational environment, it is often desirable for conflict resolutions to be consistent with objectives for metering the flow of aircraft to airports. Integrated metering and conflict resolution might be viewed as planning the satisfaction of multiple goals. The resolution of multi-conflict situations might also be structured as a multiple goal problem. Therefore AIRPAC ought to exert more explicit control over the formulation, coordination and satisfaction of goals.

As its development continues AIRPAC will be used as a tool for gathering conflict resolution expertise from air traffic controllers. A greater understanding of the conflict resolution decision process is needed to help the FAA identify the compability required of an automated resolution function for the AERA system. It has also been suggested that a knowledge-based system similar to AIRPAC might be developed for off-line training of new controllers in the use of standardized techniques for resolving conflicts and other ATC problems. A knowledge-based training system could represent decision rules-of-thumb in a form readable by controllers and could provide explanation for its decision. Such a system would provide many benefits for controller training, irrespective of whether a knowledge-based decision aid is available in the ATC operational environment.

Acknowledgment

The author wishes to acknowledge the significant controbution of Karl B. Schwamb as co-developer of the AIRPAC system. While the author focused on understanding and formulating the problem domain expertise, Mr. Schwamb was primarily responsible for creating, designing and successfully implementing the hierarchical approach to planning conflict resolutions.

References

1. Federal Aviation Administration, "The AERA Concept", FAA-EM-81-3, March 1981.

2. R. B. Wesson, "Problem-Solving With Simulation in the World of an Air Traffic Controller", Ph.D. Dissertation, University of Texas at Austin, December 1977.

3. S. E. Cross, "Qualitative Reasoning in an Expert System Framework", Report T-124, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, May 1983.

4. C. A. Shively and K. B. Schwamb, "A Knowledge-Based System for Aircraft Conflict Resolution", MTR-83W100, Mitre Corporation, McLean, Va., December 1983.

5. C. A. Shively and K. B. Schwamb, "AIRPAC: Advisor for Intelligent Resolution of Predicted Aircraft Conflicts", MTR-84W164, Mitre Corporation, McLean, Va., October 1984.

6. M. Minsky, "A Framework for Representing Knowledge", in The Psychology of Computer Vision, Edited by P. H. Winston, McGraw-Hill, New York, N.Y., 1975.

7. K. B. Schwamb, "IRIS: An Integrated Representation and Inferencing System", MTR-84W165, Mitre Corporation, McLean, Va., September 1984.

8. R. Roberts and I. Goldstein, "The FRL Manual", MIT AI Laboratory, AI Memo 409, Cambridge, MA, 1981.

9. J. Foderaro and K. Sklower, The Franz Lisp Manual, Regents of the University of California, April 1982.

10. Symbolics Inc., Lisp Language Manual, Cambridge, MA. 1984.

POTENTIAL APPLICATIONS OF ARTIFICIAL INTELLIGENCE TO THE AIR TRAFFIC CONTROL SYSTEM

Stephen M. Alvania, Federal Aviation Administration

These comments concern the potential applications of artificial intelligence (AI) to the Federal Aviation Administration's (FAA) air traffic control (ATC) system. Artificial intelligence and expert systems technology are clearly a "leading edge" in advanced computer science and must be thoroughly examined for possible benefits for the FAA and, by extension, system users.

A logical approach would be to avoid concentrating great amounts of time, money, or energy on exploring generic or high-level abstract concepts but rather to attempt to demonstrate the operational feasibility of simple, straightforward, and/or "intuitively obvious" applications. While this may not be fully satisfying to enthusiastic theorists, exotic theories remain pure fantasy until the soundness of fundamental capabilities can be shown. The following is a listing of areas that the FAA should explore.

Severe Weather Detection/Prediction

The eventual implementation of terminal Doppler weather radar systems is not an invalid assumption. Research into expert system analysis of Doppler radar data has shown that gust front and microburst activity can be automatically detected. Additional work should be done to determine the feasibility of: (a) reducing the data processing time; (b) having a capability to project the above wind shear conditions; and (c) developing a scheme for providing that wind shear data to appropriate control personnel.

Traffic Flow Management

National flow management is largely a data management and non-tactical ATC process, utilizing a relatively stable set of logical cause and effect rules. The pure enormity of the national flow management process, due to the large number of destination points, departure points, congestion points, and shifting (yet inter-related) demand levels, would appear to make an expert system application "intuitively obvious". Given the economic benefits available through a more efficient national flow management process, the FAA should explore this area as soon as possible.

System Maintenance Analysis

The FAA will be capable of collecting and storing great amounts of data pertaining to equipment performance and patterns through the remote maintenance monitoring system (RMMS). An expert system capability that could aid the system monitoring and maintenance personnel in analyzing the data to reduce the out-of-service time or the project system failures would be of significant benefit to FAA technicians, controllers, and system users. This is another area that should be explored.

Air Traffic Controller Training Aid

An expert system that could monitor controller training problem simulations (radar) and automatically interrupt the simulation when a "system error" occurred, explain why it happened, and provide a reasonable set of control instructions that would have prevented the error, would enhance the productivity of training personnel by providing a "self-study" practice capability for students. It could also enhance training quality by providing opportunities for more practice exercises. If sufficiently sophisticated, this same principle could be applied to teaching efficient control techniques. The benefits here also appear to be "intuitively obvious".

Tactical Air Traffic Control

In order to achieve significant controller productivity gains, a relatively high level of control responsibility will have to be transferred from the controller to the automation system. It would seem that expert system technology will be required to do that. This is certainly a long term activity, but the FAA must begin now to determine the likelihood that such a transfer is possible.

SPONSORSHIP OF THIS CIRCULAR

GROUP 1--TRANSPORTATION SYSTEMS PLANNING AND ADMINISTRATION
William A. Bulley, H. W. Lochner Inc., Chairman

Committee on Airfield and Airspace Capacity and Delay

David J. Sheftel, Chairman
Joseph D. Blatt; George J. Couluris; John W. Drake, Howard Eisner; Ray H. Fowler; Stephen J. Gross; Raymond J. Hilton; Stephen L. M. Hockaday; Everett S. Joline; James P. Loomis; James P. Muldoon; Thomas J. O'Brien; Amedeo R. Odoni; Roy Pulsifer; David A. Schlothauer; Agam N. Sinha; Gordon Y. Watada; Peter J. Zegan

Participants

Adkins, Alfred L., Federal Aviation Administration, Technical Center
Alvania, Stephen M., Federal Aviation Administration

Blake, Stephen E., Transportation Research Board
Brown, Robert H., National Aeronautics and Space Administration, Johnson Space Center

Campbell, Steven D., Massachusetts Institute of Technology, Lincoln Laboratory
Crosby, Robert W., Federal Aviation Administration

Gosling, Geoffrey D., University of California, Berkeley
Guth, Herbert J., Transportation Research Board

Hockaday, Stephen M., California Polytechnic State University

Kaul, Michael, BDM Corporation

Larsen, Ronald L., University of Maryland
Leonard, Paul, Air Transport Association of America
Loomis, James P., Battelle Columbus Laboratories

Montemerlo, Melvin, National Aeronautics and Space Administration Headquarters

Neumann, Paul J., Federal Aviation Administration

Pararas, John D., Massachusetts Institute of Technology, Flight Transportation Laboratory

Retelle, John D., U.S. Army, Defense Advanced Research Projects Agency
Robinson, Alfred C., Battelle Columbus Laboratories
Roditi, Salvador, Federal Aviation Administration

Schauffler, Peter, Transportation Research Board
Sheftel, David J., Consultant, McLean, Virginia
Shively, Curt, Mitre Corporation
Spencer, David A., Massachusetts Institute of Technology, Lincoln Laboratory
Swetnam, George F., Mitre Corporation

Wood, Peter, Mitre Corporation
Wright, Richard D., U.S. Department of Transportation, Transportation Systems Center