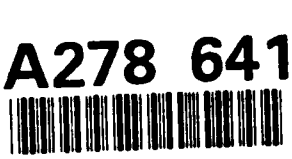


University of Southern California

Intelligent Automated Agents for Flight Training Simulators

Randolph M. Jones
Artificial Intelligence Laboratory
University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109-2110

Milind Tambe and Paul S. Rosenbloom
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Intelligent Automated Agents for Flight Training Simulators

Randolph M. Jones
Artificial Intelligence Laboratory
University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109-2110

Milind Tambe and Paul S. Rosenbloom
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

February 1993
ISI/RR-93- 350

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1993	Research Report

4. TITLE AND SUBTITLE
Intelligent Automated Agents for Flight Training Simulators

5. FUNDING NUMBERS
N00014-91-J-1624

and

DABT63-91-C-0025

6. AUTHOR(S)
Randolph M. Jones, Milind Tambe, John E. Laird, and Paul S. Rosenbloom

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USC INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292-6695

8. PERFORMING ORGANIZATON REPORT NUMBER
ISI/RR-93-350

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT
UNCLASSIFIED/UNLIMITED

12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Training in flight simulators will be more effective if the agents involved in the simulation behave realistically. Accomplishing this requires that the automated agents be under autonomous, intelligent control. We are using the Soar cognitive architecture to implement intelligent agents that behave as much like humans as possible. In order to approximate human behavior, the agents must integrate planning and reaction in real time, adapt to new and unexpected situations, learn with experience, and exhibit the cognitive limitations and strengths of humans. This paper describes two simple tactical flight scenarios and the knowledge required for an agent to complete them. In addition, the paper describes an implemented agent model that performs in limited tactical scenarios on three different flight simulators.

14. SUBJECT TERMS
artificial intelligence, believable agents, flexible behaviors, realistic training environments, Soar

15. NUMBER OF PAGES
10

16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

Intelligent Automated Agents for Flight Training Simulators

Randolph M. Jones,[1] Milind Tambe,[2] John E. Laird,[1] and Paul S. Rosenbloom[3]

[1]Artificial Intelligence Laboratory
University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109-2110

[2]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[3]Department of Computer Science and
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90292

Abstract

Training in flight simulators will be more effective if the agents involved in the simulation behave realistically. Accomplishing this requires that the automated agents be under autonomous, intelligent control. We are using the Soar cognitive architecture to implement intelligent agents that behave as much like humans as possible. In order to approximate human behavior, the agents must integrate planning and reaction in real time, adapt to new and unexpected situations, learn with experience, and exhibit the cognitive limitations and strengths of humans. This paper describes two simple tactical flight scenarios and the knowledge required for an agent to complete them. In addition, the paper describes an implemented agent model that performs in limited tactical scenarios on three different flight simulators.

The goal of this research is to construct intelligent, automated agents for flight simulators that are used to train navy pilots in flight tactics. When pilots train in tactical simulations, they learn to react to (and reason about) the behaviors of the other agents (friendly and enemy forces) in the training scenario. Thus, it is important that these agents behave as realistically as possible. Standard automated and semi-automated agents can provide this to a limited extent, but trainees can quickly recognize automated agents and take advantages of known weaknesses in their behavior. To provide a more realistic training situation, automated agents should be indistinguishable from other human pilots taking part in the simulation.

To construct such intelligent, automated agents, we have applied techniques from the fields of artificial intelligence and cognitive science. The agents are implemented within the Soar system, a state-of-the-art, integrated cognitive architecture (Rosenbloom et al., 1991). These agents incorporate knowledge gleaned from interviews with experts in flight tactics and analysis of the tactical domain. Soar is a promising candidate for developing agents that behave like humans. Flexible and adaptive behavior is one of Soar's primary strengths, and Soar's learning mechanism provides it with the capability of improving its performance with experience. In addition, Soar allows the smooth integration of planning and reaction in decision making (Pearson

et al. 1993). Finally, Soar is the foundation for the development of a proposed unified theory of human cognition (Newell, 1990), and thus maps quite well onto a number of the cognitive issues of interest. This paper reports the results of our research in constructing an intelligent agent for an initial, simple training scenario and our efforts at supplementing the agent's knowledge in order to carry out more complex missions.

Complexities of tactical decision-making

In order to complete a tactical mission, pilots incorporate multiple types of knowledge. These include, for example, knowledge about the goals of the mission, airplane and weapon constraints, survival tactics, controlling the vehicle, characteristics of the environment, and the physical and cognitive capabilities of all of the agents taking part in the scenario. In addition, pilots use their knowledge flexibly and exhibit adaptive behavior. This includes a variety of capabilities, such as reasoning about (and surviving in) unexpected situations, adapting to new situations, learning from experience, and addressing multiple goals simultaneously (e.g., protecting a position, intercepting the enemy, and surviving). Finally, pilots integrate decision-making during a mission with split-second reactions to new situations and potential threats.

Robust automated forces that can carry out general simulated missions must address these issues, especially if the forces are to behave as humans would in similar circumstances. In addition to providing the wide range of capabilities that human pilots exhibit, intelligent agents must reflect the same types of weaknesses as humans. These include mental limitations, such as attention and cognitive load, and physical limitations, such as reduced cognitive processing under high forces (such as during a hard turn).

To capture the complex interactions between agents in a simulation, we feel it necessary for each agent to be as autonomous and intelligent as possible. Simulation via stochastic methods can capture general behaviors of groups of agents, but a more realistic simulation requires each agent to behave individually, with is own set of goals, constraints, and perceptions. In addition, if the agents are to be used for training pilots, they must be intelligent in order to provide as rich a training environment as would flying against real humans.

Requirements for an intelligent automated agent

The primary research question is how intelligent, automated agents should be implemented. A simple solution would be to attempt to create "simulation-pilot expert systems". This would involve converting knowledge about high-level tactical decision-making into a fixed rule base. The system would suggest the most appropriate action (or set of actions) based on the current status of the environment. In fact, a number of expert systems have been implemented for various aspects of tactical decision-making (e.g., Kornell, 1987; Ritter & Feurzeig, 1987; Zytkow & Erickson, 1987;).

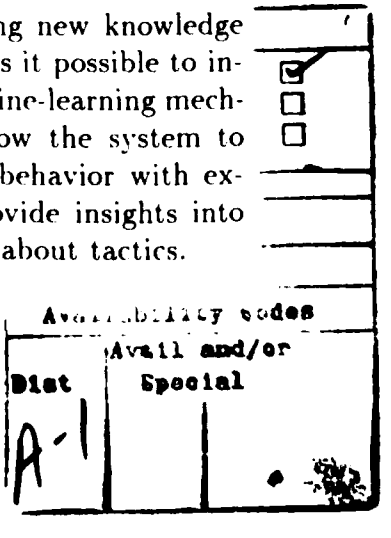
However, while expert systems have some of the strengths required for realistic simulation, they are usually weak in other areas. In a standard rule-based approach, it

is difficult to capture the complexity of the multiple, dynamic goals that pilots must reason about. In contrast, systems that *can* reason well in such a complex domain generally have difficulties making decisions in real time, and they often do not have the ability to react to changes in the environment when there is not enough time to plan ahead. In addition, systems with only high-level tactical knowledge prove to be rather rigid. Unless the system can be preprogrammed for every possible contingency, its performance degrades greatly when it finds itself in unexpected situations. Finally, expert systems generally ignore the possibility of learning with experience and other cognitive aspects of the task. Intelligent, autonomous agents must combine all of these strengths, having the ability to reason about multiple goals in a complex environment, react quickly and appropriately when the time for complex reasoning is limited, adapt to new situations gracefully, and improve its behavior with experience.

In order to create an agent that can reason and react in real time, and is flexible enough to adapt to new situations, it is not enough simply to encode high-level tactics as rules in the system. Rather, the system must also *understand* why each high-level tactical decision is made, so it must contain knowledge of the first principles that support those decisions. For example, part of one tactic for intercepting a bogey involves achieving a desired lateral separation from the bogey's flight path. One way to generate this behavior is to include a specific rule for the agent to move to the desired lateral separation when it is on the appropriate leg of the intercept. However, a more intelligent agent encodes the knowledge that explains why this particular tactic works (so that the fighter will have enough space to come around for a rear-quarter shot if the long and medium-range missiles miss).

With the appropriate supporting knowledge, the system can function in situations that the programmer may not have anticipated. Maintaining lateral separation from the bogey's flight path is a general principle that allows the fighter room to negotiate a turn for a short-range missile shot. This principle may have an impact in a large number of tactical situations, and therefore shouldn't be considered as merely an instruction to follow for one particular type of intercept. If the system reasons from first principles, the programmer does not have to hard code every possible contingency, and good variations on tactics should emerge in response to unanticipated changes in the simulation environment.

Implementing the agent in this manner also provides advantages in terms of adding new knowledge to the system. If the tactical decisions emerge from low-level knowledge, high-level decisions will change appropriately as the supporting knowledge is changed or supplemented. New low-level knowledge (such as a better understanding of geometric principles or radar limitations) will interact with existing knowledge to generate subtle (or possibly dramatic) changes in behavior. Thus, the agent can reason in a number of new situations without requiring a new specific rule for each case. The ease of adding new knowledge to the system also makes it possible to incorporate existing machine-learning mechanisms. These can allow the system to adapt and improve its behavior with experience, as well as provide insights into how human pilots learn about tactics.



The Soar architecture for problem solving (Newell, 1990) is well suited for this type of task. It divides knowledge into *problem spaces* and allows goals and actions in one problem space to be implemented via reasoning in another. Thus, when the agent has a high level goal to intercept a bogey, for example, it can switch problem spaces and reason about the characteristics of its weapons, radar, airplane, and military doctrine. The knowledge from each of these spaces combines to generate an appropriate tactical action. In turn, the high-level action can then be implemented in a problem space that contains medium-level knowledge about plane maneuvers or low-level knowledge about moving the stick and flipping switches.

Because knowledge is separated into problem spaces, it can be easily updated. For example, if the agent's plane is equipped with a new radar with a longer range, only the knowledge in the "radar" space need be updated. New decisions made in the radar space will interact with the results of reasoning in other problem spaces, eventually impacting high-level decisions such as which specific actions should be taken to intercept a bogey. Likewise, if the automated agent is moved to a new simulation environment with a new interface, we can appropriately update the knowledge in the "control" problem space, leaving the remaining knowledge intact.

Simple tactical situations

Our initial effort to construct an intelligent agent focuses on two tactical scenarios used in training pilots: the "non-jinking bogey" and "1-v-1 aggressive bogey" scenarios. In the non-jinking bogey scenario, the target is an airplane (such as a cargo or fuel plane) that holds a steady course and altitude, and does not carry any offensive threats. The key to this scenario is that the bogey does not attempt to evade (jink) the fighter's attack in any way. Although this situation is not likely to occur often in real combat situations, it is a valuable training situation for pilots. It teaches them how to line up the delivery of various types of missiles when the bogey's behavior is very predictable. When a non-offensive bogey's behavior becomes less predictable, the tactics required to intercept it actually become simpler (but less effective).

There are three main phases involved in attacking a non-jinking bogey (see Figure 1). These involve delivering long, medium, and short-range missiles. During each of the phases, the fighter must assume that the current missile will miss, and simultaneously maneuver into the most advantageous position for the next phase. For example, while moving closer to the bogey to fire a long-range missile, the fighter also attempts to achieve the best lateral separation and target aspect for a shot with the medium-range missile (see Figure 2). After delivering a medium-range missile, the fighter must perform displacement and counter turns in order to end up behind the bogey. This allows the fighter to fire a rear-quarter short-range missile. Due to these constraints, the fighter cannot simply head on a collision course with the bogey, but must get to the bogey as quickly as possible while ensuring that it can eventually achieve a rear-quarter missile shot.

The tactics for executing this scenario are relatively simple. The fighter must achieve the appropriate lateral separation and target aspect while firing its weapons at the right times. Then it must execute the dis-

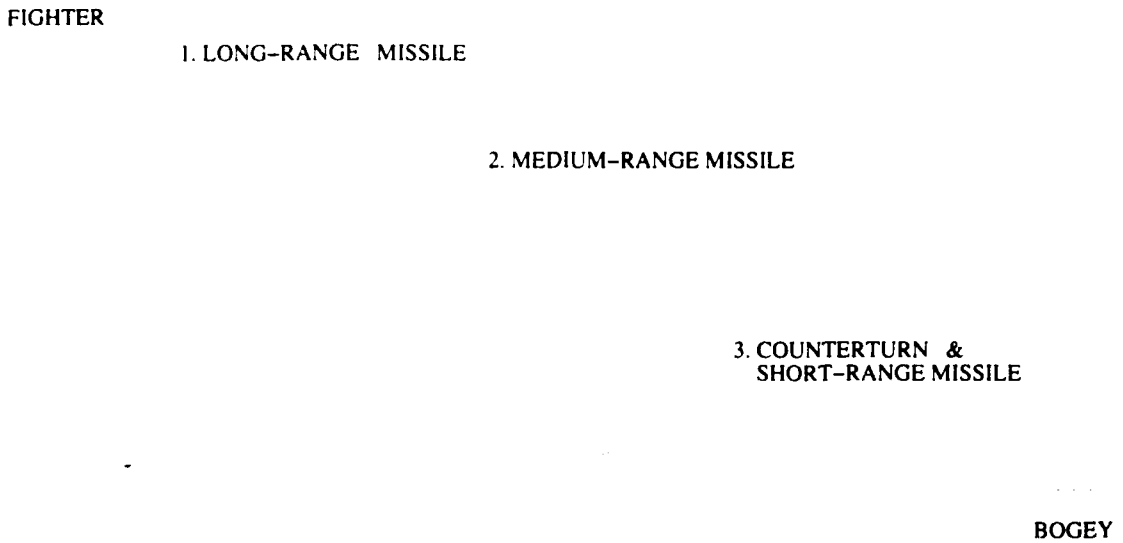


Figure 1. Three stages for intercepting a non-jinking bogey.

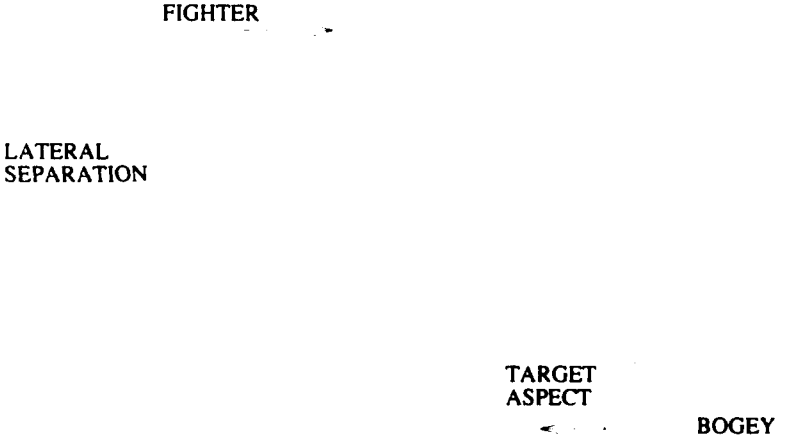


Figure 2. Definition of lateral separation and target aspect.

placement and counter turns and deliver the short-range missile. As mentioned previously, we could code these tactics directly into rules for the agent, but they would then only work under very specific circumstances where everything goes right. Thus, we have implemented the knowledge that supports these tactics. This knowledge justifies *why* each tactical decision should be made when it is made. This allows the system, for example, to get back on course for a short-range missile shot if it misses its opportunity for the medium-range missile shot for some reason. In addition, any particular action that the agent generates will be based on the supporting knowledge, and the agent has the potential to explain its decision (a facility we plan to add in the future).

The 1-v-1 aggressive bogey scenario involves two airplanes with similar capabilities. One is protecting a high-value unit and the other is attempting to destroy it. When the two fighters come in contact they both attempt to intercept and destroy each other, with the overall goal of surviving. This scenario highlights an interaction between different low-level constraints that results in tactical decisions. For example, if one fighter is equipped with a slightly better radar, missiles with longer range, or a more mobile airplane, it dramatically affects the actions that should be taken in completing the mission and surviving. Our agent so far only partially implements this 1-v-1 scenario, and it involves a number of issues that make it more complex than the non-jinking bogey scenario. After discussing the current state of the agent model, we will describe these issues in detail.

Details of the intelligent agent

In order to construct an agent that successfully intercepts a non-jinking bogey, we analyzed tactics for the scenario and interviewed former pilots and radar intercept officers. This allowed us to determine the underlying knowledge and first principles that support the tactics. Then, we encoded this knowledge into an executable Soar system.

The Soar agent's knowledge is organized into problem spaces, each containing operators that allow the agent to reason about particular types of goals. When the agent cannot immediately carry out an action at one level, it uses Soar's universal subgoaling mechanism to move into an alternate problem space and consider methods for carrying out that action. Therefore, high-level tactical decisions are eventually implemented as medium-level maneuver actions or low-level control actions, and the agent always has multiple goals in memory that it uses to reason about and react to its ever-changing situation.

Depending on the particular simulation platform, the current Soar agent requires between 13 and 17 problem spaces to reason with; i.e., 13–17 different types of goals that it reasons about. Most of these are shown in Figure 3. The mission, protect-hvu, barcap, and intercept problem spaces encode tactical knowledge for carrying out missions and performing intercepts. The problem spaces for weapons and missiles include knowledge about specific weapons and the actions that must be performed to deliver them to a target. The maneuver and absolutes problem spaces determine the actual plane maneuvers that must be carried out to implement higher-level actions. The remaining problem spaces im-

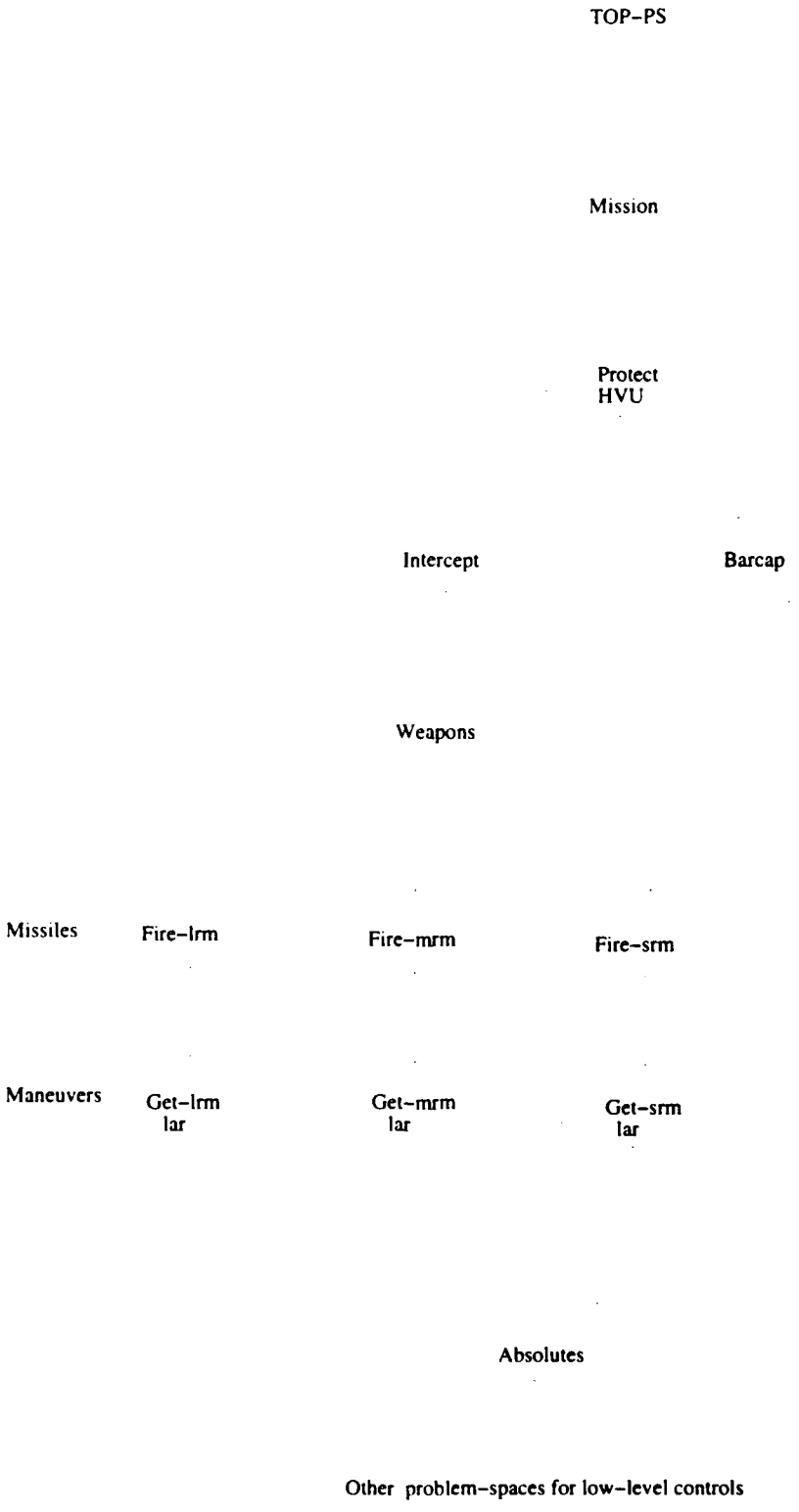


Figure 3. Soar's problem spaces for intercepting a non-jinking bogey.

plement airplane maneuvers at various levels of specification, down to the level of stick and button commands that are issued to the flight simulator.

At any particular instant, between 5 and 12 problem spaces (or hierarchical goals) are usually active. Thus, when changes occur in the agent's situation, there are multiple levels at which the agent may react (Pearson et al., 1993). For example, at a low level, a sudden down draft can cause a change in climb-rate or altitude, leading the agent directly to pull back on the stick. At a higher level, a maneuver by a bogey on the radar can cause a change in tactics. Any reasoning involved in implementing the new tactical decision also percolates down to a new maneuver or stick action. In this manner, Soar maintains its variety of goals in parallel, and violations of the goals at any level lead to immediate action at the appropriate level.

We have implemented an initial model for the non-jinking bogey scenario in whole or in part on three separate flight simulators. The simplest simulator moves planes in a two-dimensional grid-world. In addition, the planes do not move with realistic flight dynamics. We used this simulator to prototype the system and debug the high-level tactics embedded in the system. The second flight simulator was adapted from the flight simulator provided with SGI graphics workstations. It works in real time and requires the agent to issue very low level commands at the level of moving the stick (by issuing mouse pixel movements) and other low-level commands (by simulating keyboard presses). The non-jinking bogey scenario has not yet been completely implemented on this simulator, because Soar must handle the low level intricacies of simply flying the airplane as well as worrying about tactical decisions and maneuvering. Finally, we have implemented the scenario on BBN's ModSAF simulator, which has the most realistic flight dynamics of the three simulators. This simulator works in real time (with a scheduler dividing time between the simulation and agents) and it takes commands at the level of maneuver actions (such as desired heading and altitude) without making the agent concern itself with how the maneuvers are actually implemented with airplane controls.

As of now, we have not completely developed the knowledge base that would allow our agent to successfully fly the 1-v-1 aggressive bogey scenario. This scenario differs from the non-jinking bogey scenario along two major dimensions. First, the bogey maneuvers, so its behavior is not entirely predictable. Second, the bogey is aggressive and has offensive capabilities, so any action that is taken must also address the overall goal of surviving: the agent cannot simply close in on the bogey and shoot it.

In order to successfully complete a mission against an aggressive bogey, the agent must include not only extra knowledge in its tactical problem spaces, but it must also have two new capabilities to address the above issues. First, the agent must be able to interpret and assess its current situation at all (or at least most) times. This primarily involves interpreting the bogey's current actions and predicting its future actions. As with most of the agent's reasoning, the interpretation process also takes place at multiple levels. At a low level, the fighter must recognize when the bogey has initiated a turn and when it has completed one. At a higher level, the fighter

must determine whether the turn indicates some kind of threat, and what that threat may be. For example, if the bogey initially comes to a collision course with the fighter, this probably indicates that the bogey is aggressive and is going to try to shoot the fighter. If the bogey points towards the fighter and then makes a hard turn, this indicates that the bogey has probably just fired a missile. The agent must interpret the limited information it gets from its sensors. Then it must use this interpretation to predict the goals that the bogey is trying to achieve and the actions at different levels that the bogey is carrying out.

The second necessary capability for the agent is to use multiple high-level goals to constrain the actions that the agent generates. These types of goals are a bit different from the parallel goals that the Soar agent already handles, because they are not hierarchical in nature. Rather, they are distinct goals that interact with each other. For example one goal, *destroy bogey*, implies that the fighter should close in on the bogey as quickly as possible. However, another goal, *survive*, pressures the fighter to avoid the bogey in order to stay out of the bogey's weapon range. These conflicting goals both must be used to select from multiple possible actions. This type of reasoning leads directly to composite tactical actions. For example, the fighter may get close enough to fire a missile and then make a sudden hard turn. The turn must be hard enough to keep the bogey and fighter from getting close too quickly, but not so hard that the fighter loses its radar lock on the bogey (which would put the fighter at a large disadvantage). In this manner, the agent determines the best action that supports two simultaneous, conflicting goals.

The issues of interpretation and simultaneous goals are not trivial, and they play central roles in agent reasoning for any tactical situations except the simplest ones. Much of tactical decision making involves creating a model of the world from limited information and addressing multiple goals and constraints, such as the current mission, survival, and the characteristics and status of the weapons and airplane. We have not completed the incorporation of this knowledge into the agent yet, but we are taking advantage of the strengths of the Soar architecture in order to implement these two important capabilities (Covrigaru, 1992).

Discussion

We have implemented an intelligent, autonomous agent that completes missions in a simple tactical scenario. The agent is designed with flexibility in mind. It reasons from first principles about high-level tactical decisions, and is thus able to reason in unexpected situations and recover gracefully from mistakes. In addition, the agent's knowledge base is flexible enough to be easily transferred between simulation platforms and to encode new tactics in a modular fashion. We are currently implementing the knowledge necessary for the agent to complete the 1-v-1 aggressive bogey scenario. This includes addressing the two important issues of situation interpretation and achieving multiple simultaneous and interacting goals.

Our future research will involve incrementally expanding the agent's knowledge base so it can reason robustly in a wide range of 1-v-1 scenarios. We will also soon focus on modeling more complex scenarios, including those involving more than two

planes. This will also allow us to expand the agent's coverage of the cognitive behaviors involved in tactical flight. For example, we will incorporate more intelligent methods for situation assessment, modeling other agents (i.e., robustly predicting actions and goals of other participants in the scenario, both friends and foes), identifying potential threats, and reacting to them. Beyond that, we will focus on more complex cognitive tasks, such as more complete integration of planning, reaction, and execution, more sophisticated interpretation of the environment and other agents, and learning from instruction.

References

Covrigaru, A. (1992). Emergence of meta-level control in multi-tasking autonomous agents (Technical Report No. CSE-TR-138-92). Doctoral dissertation, Dept. of Electrical Engineering and Computer Science, University of Michigan.

Kornell, J. (1987). Reflections on using knowledge based systems for military simulation. *Simulation, 48*, 144–148.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Pearson, D. J., Huffman, S. B., Willis, M. B., Laird, J. E., & Jones, R. M. (1993). Intelligent multi-level control in a highly reactive domain. In *Proceedings of the International Conference on Intelligent Autonomous Systems*.

Ritter, F., & Feurzeig, W. (1987). Teaching real time tactical thinking. In J. Psotka, L. D. Massey, & S. A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Lawrence Erlbaum.

Rosenbloom, P. S., Laird, J. E., Newell, A., & McCarl, R. (1991). A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artificial Intelligence, 47*, 289–325.

Zytkow, J. M., & Erickson, M. D. (1987). Tactical manager in a simulated environment. In Z. W. Ras & M. Zemankova (Eds.), *Methodologies for intelligent systems*. Amsterdam: Elsevier Science.