Constraints on the Design of a High-Level Model of Cognition

Randolph M. Jones (RJONES@EECS.UMICH.EDU)
John E. Laird (LAIRD@EECS.UMICH.EDU)
Artificial Intelligence Laboratory
University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109-2110

Abstract

The TacAir-Soar system is a computer program that generates human-like behavior flying simulated aircraft in tactical air combat training scenarios. The design of the system has been driven by functional concerns, allowing the system to generate a wide range of appropriate behaviors in severely time-limited situations. The combination of constraints from the complexity and dynamics of the domain with the overall goal of human-like behavior led to a system that can be viewed as a model of cognition for high-level, complex tasks. This paper analyzes the system in such a light, and describes how the functional design constraints map on to cognitively plausible representations and mechanisms, sometimes in surprising ways.

Introduction

For the past few years, we have been developing a computer system, called TacAir-Soar, that "flies" aircraft in tactical air combat simulations (Tambe et al., 1995). The overall goal of this work is for the system to generate behavior that looks like it is being generated by an expert-level human. The evaluation of our system takes place at a very high level of behavior. To test TacAir-Soar, we place it in a variety of different situations, flying different types of combat aircraft, with different types of missions and different combat situations. We then let the simulation run while military experts observe the behaviors exhibited by the system's aircraft. At this level of observation, the experts do not have access to the step-by-step reasoning of the system. Rather, the experts observe aircraft maneuvers (such as changes in altitude, heading, and speed), employment of weapons, and communication between different agents in semi-natural language. The experts interpret these observable operations in terms of how they fit into even higher-level behaviors, such as specific weapons-employment tactics or progress toward achieving mission goals. If the behaviors fit a plausible overall plan that a human expert would execute, they are judged to be correct.

In this paper, we explore the ways in which TacAir-Soar can be considered a psychological model. Given the high level of observable behavior, it could be argued that TacAir-Soar is at best a knowledge-level model of cognition (Newell, 1982). Just because the system exhibits appropriate behavior at a gross level of observation, does not necessarily indicate that the processing within the system is anything like what a human does on the same task. Thus, it can be argued that generating appropriate high-level behaviors is a very weak constraint on the design of the cognitive model.

However, the design of TacAir-Soar embodies at least two other types of constraints, involving the characteristics of the air combat task and the characteristics of the system's perceptual-motor components. We argue that these constraints have been sufficient to force the design of TacAir-Soar towards an accurate symbol-level model of human expert behavior in air combat. Additionally, there is a third constraint on the design of TacAir-Soar that has little to do with psychological concerns. The TacAir-Soar system will be used in a military exercise simulation, called STOW, in late 1997. Current plans are for the TacAir-Soar model to control over 200 different aircraft at the same time. Thus, a practical constraint on the system design is for it to be as efficient as possible, so that fewer computers will be necessary and the exercise will be as cheap as possible. Our initial expectations were that we would have to compromise our desire to simulate human behavior faithfully in order to meet the efficiency requirements. However, we have been surprised to discover that every time we make a design change for efficiency, it appears to increase the psychological plausibility of the model as well. This paper describes these constraints, as well as some specific details of how they influenced the design of the TacAir-Soar system. We argue that because TacAir-Soar includes these constraints in its design, the system can be used to make claims about how human pilots must be reasoning.

The Architecture of the Model

The simulation of human pilot behavior consists of four major components:

1. The external environment. In this domain the environment is a distributed simulation system that implements realistic models of various aircraft, weapons, and sensors.

2. The input system. This is an interface that describes (a portion of) the current state of the environment at some level of representation.

3. The agent. The agent is a human or simulation model (in our case, TacAir-Soar), which receives information about the environment from the input system, and uses some reasoning process to generate expert-level external behaviors.

4. The output system. This is the interface through which the agent can take action in the environment. All agent decisions must ultimately lead to output-system commands if the agent is going to have any influence on the environment. As with the input system, there are many possible levels of representation for the output system.

Given this structure, we can now more clearly define the task of this research. Our goal is to understand how human combat pilots reason and perform their tasks. To this end, the model that implements the agent can be viewed as a dependent variable. The external environment, input system, and output system are the independent variables. The characteristics of each of these systems impose constraints on the design of the agent model. If we "clamp" the design of these three systems to simulate actual air combat as closely as possible, and if the agent model is able to generate human-like high-level behavior, then the reasoning processes in the agent provide a hypothetical model of the reasoning processes of a human pilot. We can use the agent model to make predictions about human pilot behavior. In addition, we can examine changes to the input and output systems that lead to improved behavior in the model, and use these to propose potentially beneficial changes in the interfaces used by human pilots. As we have mentioned above, many of the changes we have considered and implemented arise from the additional constraint of building as efficient a simulation system as possible. We discovered that changes that improve the efficiency of the overall simulation architecture generally also improved the simplicity and quality of the agent model.

Constraints on Cognition

We are now in a position to examine the constraints on cognition in the tactical air combat domain. As suggested above, we can divide this discussion into three components. After we have determined how to make the environment, input system, and output system as realistic as possible, we can analyze their influence on the agent model. We will also analyze the impact of intermediate versions of the agent model on subsequent versions of the environment simulation, input system, and output system.

Environmental Constraints

The constraints arising from the simulation environment consist of characteristics of the task domain together with the characteristics of the underlying simulation of the environment. As we have discussed, the task domain is tactical air combat, with the general goal of flying missions in the same way that expert humans do. Successfully flying air combat missions requires a fair amount of knowledge and expertise, and there are a number of complex concepts and tradeoffs that must be taken into consideration at any given point during a mission. This implies that the agent model must at least contain a large amount of knowledge, although it does not necessarily specify a particular representation for that knowledge. As an indication of the amount of knowledge required, the current version of TacAir-Soar consists of over 4500 production rules, implementing more than 400 operators.

Additionally, the primary sources of information we consult in building TacAir-Soar come from the United States military. These sources (experts and documents) represent doctrine, missions, and tactics in terms of hierarchical goals. Because we are trying to generate behavior similar to the military, this implies a constraint on the actual representation of knowledge withing the agent model. This is especially true if the agent model is going to interact with real humans (such as flying missions in groups).

Aside from the complexity and representation of knowledge, a key characteristic of air combat is that the situation is always changing, and usually quite rapidly. Depending on the situation, human pilots must make decisions under severe time constraints. Naturally, there is often an inverse relationship between the time available to make a decision and the quality of generated behavior. However, it is conceivable that certain types of decisions that humans are slow to make can be computed quickly by computer simulations. Thus, as part of our goal of mimicking human behavior, we should watch for the possibility of the agent model being "too good" in some situations. This is a further constraint imposed by the domain, although it is tempered somewhat by the "meta-constraint" of designing as efficient a simulation system as possible.

Additional constraints on reasoning come from the simulation of the air combat arena. If the simulation of the world, aircraft, weapons, etc. is not realistic, it is unlikely that behaviors that would be appropriate in the real world would also be appropriate in the simulated world. Thus, if we want to be "forced" to design an accurate agent model, the simulation platform must provide a realistic environment. The TacAir-Soar system has been designed to run with the ModSAF (Calder et al., 1993) distributed simulation system. ModSAF provides air frames with realistic flight dynamics, and many of the sensor and weapon simulations are based on models of real systems that have been validated by the United States military. There is room for improvement in some of the systems, but overall the ModSAF simulator provides a quite realistic simulation of the air combat environment.

Input System Constraints

As mentioned in the previous section, if we hope to end up with a realistic agent model, it must receive its information from accurate simulations of different sensors. The sensor simulation determines when information should be made available to the agent and what the content of the information should be. However, it does not necessarily dictate a particular representation for sensor information, which will also have a significant impact on the agent model. Representation decisions are part of the design of the input system,

The choice of input representations relies a great deal on the level of cognition we want the agent model to simulate. For example, a complete model of cognition would probably incorporate a realistic vision system, and the representation of input would be very low-level visual primitives. Even with-

out a high-fidelity vision model, it might be desirable to model the tracking of eye movements and focus of attention. For the initial design of TacAir-Soar, however, we made the decision to represent input at a relatively high level of symbolic concepts. TacAir-Soar is supposed to model expert-level behavior, and we felt safe in assuming that expert pilots have well rehearsed the routines for parsing visual cues, understanding radar symbology, etc. Thus, the agent model does not focus on behavior at that low a level. It should also be made explicit that choosing a relatively high level of representation allowed us to build a more computationally efficient system.

It is worth going over TacAir-Soar's input system in some detail, to provide concrete examples of the input system representation. In the tactical air combat domain, there are three general sources of information from the environment:

1. Communicated information coming in over the radio.
2. Various types of information from cockpit displays and gauges, such as altimeter readings, and radar contact information.
3. Visual contacts acquired by looking out the cockpit canopy.

The first of these has the most simple representation in TacAir-Soar's input system. Radio messages consist of a list of symbols representing the words in the message, together with a symbol representing which radio the message came in on (there are typically two radios in a fighter aircraft). Such a representation requires the agent model to parse strings of words in semi-natural language, but not to worry about garbled messages or faulty radios, etc.

Again, the design of the input representation depends on the level of behavior we care to model. The version of TacAir-Soar we discuss here does not flexibly understand or generate natural language, using a large set of semi-natural templates instead. However, a group at Carnegie Mellon University is developing a realistic natural language processor to integrate into the system (Lehman, Van Dyke, & Rubinoff, 1995).

It is interesting to note that the military uses template-like "comm brevity" codes in training their pilots, but no human ever sticks to the rules during a real combat situation. Thus, the current agent model is adequate for agent-to-agent simulated communication, but it breaks down when communicating with a human.

The representation of input from most of the cockpit displays is also relatively straightforward. For example, an altimeter in a real aircraft would consist of two hands pointing at numerals on a dial. TacAir-Soar's input system simply represents this information with a numeral. Examples of other quantities represented with simple symbols include the aircraft's speed, heading, and roll, as well as the type of the currently selected weapon (which appears in the heads-up display of a real aircraft).

Visual contacts acquired through the cockpit canopy and radar contacts appearing as blips on a radar display have similar, and somewhat more complex, representations for TacAir-Soar. Each contact is represented by a symbol, and this symbol in turn has a set of associated symbols describing the contact. The associated symbols can include things like the contact's altitude, heading, and range, and the contact's aircraft type. The initial implementation of the input system reported a small number of quantities for every existing contact (i.e., every entity that could be seen out the canopy or that appeared on the radar scope). However, as we developed TacAir-Soar's agent model, there were some interesting interactions between design decisions for the agent and the input system. We will discuss some of these later.

Output System Constraints

As with the input system, the initial design of the output system reflects as much as possible the types of devices real humans use to fly air combat missions. This includes controls for flying the aircraft, buttons and switches for controlling the weapons systems, and buttons to control the radar display.

Also similar to the input system design is our choice to represent output actions at a level consistent with the level of behavior we are interested in modeling. Once again, because we are modeling expert humans, we can assume that most control actions are well rehearsed, so we do not have to model output at the level of muscle control or physical movements. The output system design assumes, for example, that all it takes to press the fire button is the explicit intention to press the button, as determined by the agent model for behavior.

Likewise, our initial design assumed that the agent model could control the maneuvers of the aircraft at a fairly high level. The output system accepts commands from the agent to set a desired speed, heading, and altitude. The actual combination of aileron, elevator, and thrust parameters required to satisfy the desired maneuver are computed by the underlying simulation platform for the aircraft. The design of the output system evolved quite a bit from its initial conception to the present. As we will discuss later, certain weaknesses in the early agent model required us to add extra levels of functionality to the output system. These enhancements in turn imposed new constraints on the agent model, focusing it more closely to an accurate model of human behavior.

The Agent Model

With all of the above design decisions, we were able to specify the task constraints, the details of the environmental simulation, and the specifics of the input and output systems. At this point we were ready to design the agent model. At the knowledge level, all the agent model needs to do is map information from the input system to commands issued to the output system in order to generate appropriate high-level behaviors. However, as we have suggested, all of the constraints discussed above combine to specify in more detail what properties the agent model must have.

The TacAir-Soar model is implemented within Soar, a computational architecture for simulating cognition (Newell, 1990), so all knowledge is specified in the form of condition-action rules. The rules match against symbols from the input

system as well as internally created symbols, and fire to create new internal symbols (such as subgoals or conceptual interpretations of the environment) or to send commands to the output system in order to generate external behavior.

The design of the rule base was immediately influenced by all of the constraints mentioned above. To begin with, the rules must interface with input and output systems that match real cockpit interfaces very closely. In addition, output commands control systems that are simulated to behave as realistically as possible. This means that the rules must not only generate the right behavior in the right situations (i.e., function at the knowledge level), they must map input symbols that correspond roughly to an expert pilot's high-level perception to symbols that correspond roughly to an expert pilot's high-level action. Thus, the realistic design of the input and output systems takes a large step in constraining the agent model to be a symbol-level model of expert pilot behavior.

Furthermore, the constraints of the task had a great influence on the representation of knowledge in the agent model. As discussed previously, the tactical air combat domain requires the flexible and often rapid generation of appropriate behavior in a complex environment, as well as a general hierarchical representation of concepts. This led us to represent knowledge as a combination of a bottom-up hierarchy of interrupt-driven rules together with a top-down goal hierarchy that focuses the context of interrupt processing.

This is perhaps best illustrated with a simplified example. Suppose the agent has decided to intercept and shoot down a target denoted by a radar blip on the agent's screen. The agent might have active hierarchical goals of destroying the target, employing weapons against the target, and shooting a sidewinder missile at the target. Now suppose the radar indicates a change in heading by the target. A bottom-up rule might calculate a new attack heading for employing the missile. Another rule would combine this result with the goal of using a sidewinder, to determine that the new geometry is outside the range of a sidewinder missile. This might lead another rule to propose using a longer range sparrow missile instead.

The combination of top-down and bottom-up hierarchies directly addresses the dual constraints of rapid behavior generation in a knowledge-rich domain. As new information comes from the input system, it leads to changes in the bottom-up hierarchy, but the hierarchy is designed so that changes only occur at the most operational level as defined by the goal hierarchy. In the previous example, if the target radar blip only changed heading by a little bit, it would not necessarily demand a new computation of attack heading. Alternatively, a new attack heading might be necessary, but the target geometry will still allow a sidewinder missile shot. If the target changed heading enough to assume a highly defensive posture, a larger portion of the bottom-up hierarchy might be recomputed, leading to a change in higher-level goals. For example, the agent may decide to employ some new tactic that takes advantage of the target's defensive posture.

Testing and Learning from the Agent Model

Once the agent model was designed and implemented, we were in a position to do two things. First, we needed to test the model by putting it into various combat situations and getting feedback from expert military observers. Second, we were able to profile the execution of the agent model in order to determine which types of reasoning were leading to overwhelming amounts of processing, suggesting areas that we might improve the efficiency of the simulation. We used feedback from both of these processes to refine the agent model, bringing it closer to our ideal model of expert behavior.

Perhaps more importantly, our experiences in testing the agent model led to quite a few changes in our design for the input and output systems. The following sections describe and justify some of the design changes we have made as we have continued development and testing of the system.

Automated Spatial Perception

In our initial input system design, radar blips and visual contacts were described with just a few attributes, indicating altitude, heading, speed, and range. This design corresponded to our impression of what information was immediately available to a human on the radar screen. However, when implementing high-level tactics, such as the intercept of a target, the agent must have access to various higher-level types of spatial and geometric information, such as the collision course to the target, the amount of angle the target is facing away from the agent, the separation between the agent and the target's flight path, and many others. Each of these quantities are used for some part of tactical decision making.

Because this level of information is necessary for each contact, the agent had to compute it from the information provided by the input system. This was a time-consuming process that required little "intelligence". Because this seemed like a behavior that could be easily automated, and because of our goal of having a very efficient simulation, we moved the computation of these higher-level attributes from the agent model into the input system.

The input system is written in the C programming language, and is very efficient at these computations because they merely involve various arithmetic operations and function calls. Computing the quantities in the agent model incurred significant overhead from matching rules against the input system and then creating new memory elements when the rules fire. This re-design of the input system led to large improvements in performance. The fact that these enhancements reduce the cognitive load of the agent model predict that similar enhancements to real radar interfaces could prove quite beneficial to human pilots. In fact, after redesigning the simulation, we discovered that some of these capabilities do exist in a few very modern radar systems. This further validates the re-design of the input system and agent model.

Focus of Attention

Even after moving computation from the agent model into the input system, there were still times when the agents could get

bogged down processing input information. Another problem we encountered was that the system would slow down as the number of contacts an agent could see increase. This occurred because the input system was computing all the input values for every contact, regardless of whether the information was actually useful. The design basically followed the principle, "more information is better," without regard to the cost of acquiring that information.

There are many different types of contacts in tactical air combat, and one of the hardest tasks in the domain is sorting these types out. Some contacts represent friendly forces coordinating with the agent. Others are friendly forces flying unrelated missions. Some contacts are hostile aircraft that must be intercepted by the agent, and some are hostile aircraft that someone else will take care of. Finally, most contacts are initially of an unknown nature, until they can be identified as friendly or hostile somehow.

Contacts of each type require different types of processing. For example, an agent need not concern itself with detailed position information of most other friendly forces. Only moderate attention must be paid to hostile forces that are far away or assigned to someone else. On the other hand, very careful attention must be focused on a target contact. It should be clear that these different types of contacts provide an additional opportunity for improving the efficiency of the system.

We built knowledge into the agent model to classify contacts into three broad categories, demanding low, medium, and high levels of attention. We then built new commands into the output system, allowing the agent to designate an attention level for any radar or visual contact. Finally, we changed the input system so that it would only compute as much geometric information for each contact as was required by the specified attention level.

These changes incurred a small overhead in processing in the agent model, but this was more than compensated by improvements in the efficiency of the input system. What pleasantly surprised us about these changes was that they improve the human-like qualities of the agent model, even though they were driven mostly by the need for computational efficiency. It is clear that humans use *some* kind of mechanism for focusing attention, although it probably has significant differences from TacAir-Soar's focusing mechanism. However, this provides a solid computational argument for why focus of attention is so important to an intelligent system.

Wider Range of Aircraft Control

For most of the missions that TacAir-Soar flies, we found the initial model for aircraft control sufficient. Recall that this involves issuing commands to the output system to set the desired heading, speed, and altitude of the aircraft. The initial design of the output system also allowed the agent to set a turn rate for changes in heading. We did find, however, that some tactical behaviors require a finer level of control.

For example, the initial control interface makes it easy to line up missile shots in two dimensions. The agent simply computes an appropriate attack heading, issues a command for the aircraft to come to that heading, waits for the desired heading to be achieved, then fires the missile. However, air combat takes place in three dimensions. Especially for close-range shots, the agent must take altitude differences into account by setting an appropriate pitch for the aircraft.

Another case arises for aircraft delivering bombs to a ground target. One tactic for delivering bombs is to fly toward the target at low altitude, then "pop up" to acquire the target visually, and then drop the bombs. Improved aircraft control aids this tactic in a number of ways. First, when the agent initiates the pop-up maneuver, it must do so by specifying a particular pitch for the aircraft. Next, the agent can only acquire visual targets through the simulated canopy, which is on the top of the aircraft. This requires the agent to roll the aircraft 180 degrees. Finally, when dropping bombs, an aircraft will have much greater accuracy if it is diving toward the target than if it is flying straight and level. This again requires control over the pitch of the aircraft.

None of these special maneuvers can be executed by the simple, initial control system provided by the ModSAF flight simulator. Thus, we needed not only to improve the output system, but the underlying simulation as well, to allow these maneuvers to be executed. This is an instance where certain behaviors in the domain required us to make the environmental simulation even more faithful to the real world. However, from the cognitive perspective, our success with the simple output system for many tactics suggests the advantages of introducing new simplified interfaces to the control of real aircraft. As it turns out, many modern commercial aircraft do have such simplified, high-level interfaces. It is an empirical question whether such interfaces would also benefit fighter pilots, or whether the low-level interfaces help them to stay alert and to feel "in control".

Automated Memory Aid

The final enhancement we will discuss arises once again from our concern for efficiency in the overall simulation. The initial agent model had one other task that would sometimes overwhelm the cognitive processing. When the agent acquires a new visual or radar contact, it creates an internal representation of that contact. Then, if contact is lost, the agent will at least remember that someone was out there, and the agent can attempt to reacquire contact or take appropriate action. However, the lost contact is certainly still moving, and in addition the agent may need to keep track of higher-level attributes (collision-course, target aspect, etc.) for the contact. The earlier agent model computed all of this "projected" information itself, because it could not be provided by the input system when the contact disappeared.

These computations became significant especially when a number of current contacts would disappear at the same time, for example, if the agent took its aircraft into a defensive maneuver. The agent's processing would suddenly grind down while it updated geometric position information for each of the missing contacts. The information computed by the agent was exactly the same information that would be provided by

the input system's radar model, if the radar contact were active. Thus, we decided once again to move this processing from the agent model into the input system.

We accomplished this by introducing a new command in the output system. This command allows the agent to create a "fake" radar blip with an initial speed, heading, altitude,bearing and range. The new input system then processes this blip by updating its position periodically and then computing the same attributes (at the appropriate attention level) it would compute for a normal radar contact. This ability leads to vast improvements in efficiency in certain situations. Once again, our success with building this capability into the TacAir-Soar system suggests possible enhancements to the radar interface of real combat aircraft.

Learning from the Agent Model

There were also times when the agent model exposed gaps in our knowledge of the domain, and helped direct knowledge acquisition efforts. An example concerns the development of behaviors for delivering bombs to a ground target. Our initial design used visual contact information for the ground target, with the agent model computing possible bomb trajectories in an effort to decide when bombs should be dropped. This was a very expensive reasoning process, much like the computations the agent model initially performed for airborne contacts. Thus, we naturally pursued the course of moving these computations into the input system.

After struggling for a few days with a design for the new input system, we discussed the matter with military experts. They informed us that combat aircraft already have such an interface built in. The interface is called a Computer Controlled Impact Point (CCIP), and it performs exactly the function we were trying to build into our own input system. Thus, we were able to solve this problem by building our own version of the CCIP, based on the interface in existing combat aircraft. This episode had some frustrating aspects, but it was reassuring that the constraints on the agent model guided our knowledge acquisition efforts in a productive direction.

Discussion

We have presented a system designed to exhibit high-level human-like behaviors in a complex domain. On the surface, this seems like a knowledge-level task, but we have presented a number of features of the particular simulation domain that tightly constrain the behavior model. We argue that these constraints are sufficient to view TacAir-Soar as a symbol-level, cognitive model of high-level human behavior for the tactical air combat domain. Most of the constraints have to do with making the agent model's environment as realistic as possible. However, we were surprised to discover that the additional "engineering" constraint of building a highly efficient simulation system led directly to improvements in the quality and accuracy of the cognitive model.

As further evidence that the TacAir-Soar agent provides a good model of human cognition, we have discussed a number of enhancements to the overall simulation design as development of the model progressed. Each enhancement maps quite plausibly onto the real domain of air combat, and the model thus makes reasonable predictions about human behavior and the effects of modifications to some aircraft interfaces.

Future work with TacAir-Soar will expand and improve the behaviors generated by the system, as well as continuing our search for a more efficient simulation system. We expect that further efficiency enhancements will make even more suggestions about potential modifications to combat aircraft interfaces. In addition, we hope to use the model to explore how expert pilots think and reason about their domain, with the hopes of identifying better representations and reasoning processes. Further down the road, we plan to use the existing expert model of behavior as a target for a system that acquires expertise automatically. The learning model will be even more constrained by human behavior, and will hopefully tell us more about how humans learn, reason, and interact with the world.

Acknowledgements

The initial design of TacAir-Soar was developed and implemented by the authors together with Milind Tambe, Paul Rosenbloom, Frank Koss, and Karl Schwamb. Paul Nielsen has also provided significant input to the current TacAir-Soar system for flying simulated fixed-wing aircraft. Paul Rosenbloom's group at the Information Sciences Institute of the University of Southern California is now working on a parallel effort to build a similar system to control simulated rotary-wing aircraft. Although we have presented the fixed-wing version of the system for this paper, the constraints and contributions of the rotary-wing version are similar.

References

Calder, R. B., Smith, J. E., Courtemanche, A. J., Mar, J. M. F., & Ceranowicz, A. Z. (1993). ModSAF behavior simulation and control. *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: University of Central Florida Institute for Simulation and Training.

Lehman, J. F., Van Dyke, J., & Rubinoff, R. (1995). Natural language processing for IFORs: Comprehension and generation in the air combat domain. *Proceeding of the Fifth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: University of Central Florida Institute for Simulation and Training.

Newell, A. (1982). The knowledge level. *Artificial Intelligence, 18*, 87–127.

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

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. B. (1995). Intelligent agents for interactive simulation environments. *AI Magazine, 16*(1), 15–39.