Toward Cognitive Robotics*

John E. Laird
Computer Science and Engineering
University of Michigan, 2260 Hayward St. Ann Arbor, MI, USA 48109-2121

ABSTRACT

Our long-term goal is to develop autonomous robotic systems that have the cognitive abilities of humans, including communication, coordination, adapting to novel situations, and learning through experience. Our approach rests on the recent integration of the Soar cognitive architecture with both virtual and physical robotic systems. Soar has been used to develop a wide variety of knowledge-rich agents for complex virtual environments, including distributed training environments and interactive computer games. For development and testing in robotic virtual environments, Soar interfaces to a variety of robotic simulators and a simple mobile robot. We have recently made significant extensions to Soar that add new memories and new non-symbolic reasoning to Soar's original symbolic processing, which should significantly improve Soar abilities for control of robots. These extensions include episodic memory, semantic memory, reinforcement learning, and mental imagery. Episodic memory and semantic memory support the learning and recalling of prior events and situations as well as facts about the world. Reinforcement learning provides the ability of the system to tune its procedural knowledge – knowledge about how to do things. Mental imagery supports the use of diagrammatic and visual representations that are critical to support spatial reasoning. We speculate on the future of unmanned systems and the need for cognitive robotics to support dynamic instruction and taskability.

Keywords: Cognitive robotics, cognitive architecture, AI, reinforcement learning, episodic memory

1. INTRODUCTION

Concurrent with the continued progress in low-level perceptual and control systems for robotics has been progress in cognitive architecture[1] – the fixed structures that support complex cognitive behavior. The time is now ripe to explore the merger of these fields into *cognitive robotics*[2, 3, 4, 5, 6, 7], whose goal is to develop unmanned systems that not only have the ability to interact with physical environments, but also have the cognitive capabilities normally associated only with human behavior. For unmanned military systems, these capabilities include:

1. Integrating perceptual and motor systems with cognition
2. Encoding large bodies of knowledge about the world, including appropriate doctrine and tactics
3. Reacting in real-time to unexpected changes in the environment
4. Executing appropriate doctrine and tactics for the current mission and tactical situation
5. Planning future behavior that is consistent with the current mission and established doctrine and tactics
6. Replanning as the situation or mission changes
7. Anticipating the actions of others
8. Communicating and coordinating behavior with commanders and teammates, both robotic and human
9. Adapting to the environment
10. Learning new missions, doctrine, and tactics from human commanders

Although one could attempt to develop cognitive robotic systems using standard programming languages, one is inevitably faced with the difficulty of how to represent, execute, and learn a wide variety of knowledge for many different tasks in a consistent and timely fashion. Approaches such as finite state machines and MDPs work well for specific tasks were there are limited and predictable numbers of features and thus not an overwhelming number of states; however, they do not scale to complex behaviors in open environments, or situations where all of the features of the task are not known ahead of time.

*laird@umich.edu; phone 1 734 647-1761; http://ai.eecs.umich.edu/people/laird/

Although there are specialized architectures and algorithms that are appropriate for specific problems, what distinguishes general intelligent entities is their ability to solve not just a single problem using a specific method, but the ability to pursue a wide variety of goals embedded in many different problem spaces and to use their knowledge in many ways – to assess the current situation in the environment, to react to changes in the environment, to deliberately select actions in order to pursue goals, to plan future actions, to reflect on past behavior in order to improve future performance, and to adapt to regularities in the environment. Cognitive architectures attempt to provide the primitive computational resources for developing intelligent systems – the primitive memories, processing units, representations, and interfaces from which more advanced cognitive capabilities are constructed. Some of them are also designed so that they meet the constraints of real-time behavior, and scale to large knowledge bases without losing reactivity. Thus, the same architecture is used across all tasks. Domain knowledge is used to specialize behavior to specific tasks and missions so that a single agent, encoded with knowledge for many domains, can pursue a variety of tasks and missions, and with learning, it can dynamically extend the tasks it performs. Each new task or domain does not require starting from scratch – task knowledge must be added, but the basic structure and general knowledge is shared across all tasks. An architecture also provides the focal point for integration of research on perception, control, motor systems and cognition, which includes the cognitive capabilities needed to generate basic doctrinally and tactically appropriate behavior. Research into more advanced capabilities, such as coordination, metacognition, and advanced learning mechanisms builds on those core capabilities.

4D/RCS[8] is one example of a middle ground between cognitive architecture and robot control systems, as it provides a framework for the development of large scale, hierarchically organized doctrine and tactical knowledge. However, to date it has not tackled many of the additional capabilities, such as learning, that has become a hallmark of the cognitive architecture approach.

Our approach to cognitive robotics is grounded in over twenty-five years of research on cognitive architecture in which we have developed the Soar cognitive architecture[9]. The Soar cognitive architecture is unique in that it is industrially robust and it has been used for building complex, knowledge-intensive agents, including controlling real-time simulated fixed-wing and rotary-wing aircraft[10], natural language processing[11], modeling air traffic controllers[12], and modeling adversaries in urban combat controlling characters in virtual environments[13]. For example, TacAir-Soar[10] models human tactical behaviors across a wide-spectrum of USAF missions, generating behavior in real-time, flying missions in coordination with both humans and synthetic agents. In addition, Soar matches the basic structure of human cognition, which is important when supporting human-robot interaction. By "thinking the way people think," Soar makes it easy for people to build internal models of its operation and predict its behavior, critical for effective human-robot interaction.

Many years ago, we made a foray into cognitive robotics and interfaced Soar to simple robots[14, 15]. Although we demonstrated that Soar could support many of the cognitive aspects required for cognitive robotics, it was clear that the next important steps had to be in improved perception systems. This led to us abandoning robotics and instead interfacing Soar to complex simulation environments[10, 16, 17]. Much has changed in the ensuing years, including advances in perception and control, but also in cognitive architecture, and specifically Soar. Others have also explored integrated cognitive architectures with robotic systems, including ACT-R[4, 5, 6], and earlier versions of Soar[2, 3, 11]. We have once again interfaced Soar to robotics, both simulated and real. Although it is early in our explorations, in this paper we layout what is currently possible and a vision for the future of cognitive robotics. We start with a brief overview of our current experimental robotic platform. The main body of the paper goes through the different components of Soar, illustrating how Soar supports the required cognitive capabilities in previous systems and in our current robot when appropriate. We conclude with a discussion of the future of cognitive robotics and the need to create robots that are dynamically taskable and instructable.

2. ROBOTIC PLATFORM

Our current work in robotics uses a custom small robotic platform called the "splinterbot" developed by Professor Edwin Olson[17] as show in Figure 1. Splinterbot is a simple indoor battery-powered wheeled robot that can move forward, turn in place, or turn while moving. It has a SICK forward looking laser range finder. For development, we have run simulations of it in USARSIM, as well as a custom simulator developed for the robot (by Professor Olson). Our software is directly portable from the simulators to the robot. When controlling the robot, the Soar agent runs on a laptop that sits on the robot and interfaces through an Ethernet port. Soar receives the following input: current position, heading, and

velocity based on odometery, which is accurate enough for indoor navigation within a small room; and range data from the SICK sensor binned into five arcs.

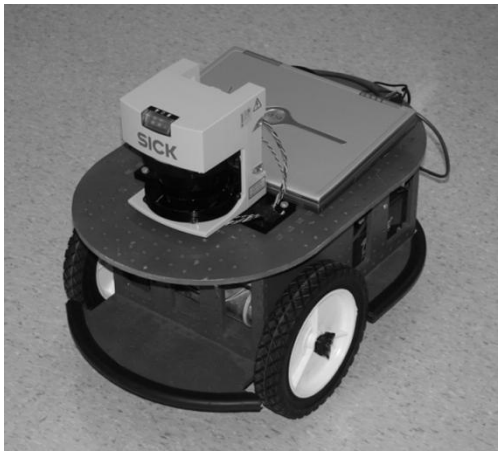

Fig. 1. Splinterbot

The current tasks that the agent performs are navigation based on waypoints (including planning and path following), dynamic obstacle avoidance, replanning when a path is blocked, station keeping, and communication with other robots about obstacles and unfriendly entities (these latter two are supported only in simulation as of now). At this point, this is not a sophisticated robotic system, but it provides a context in which we can describe how different cognitive capabilities can be supported by a cognitive architecture for a robot.

3. BASICS OF THE SOAR COGNITIVE ARCHITECTURE

In this section, we present the basics of the Soar architecture, illustrating its capabilities with previous Soar agents we have implemented in synthetic environments. All of these systems have to deal with uncertain and incomplete information, and dynamic environments. All the synthetic environments were developed by other organizations, and although they do not have all of the complexity of the real world, they are "real" environments in that they are used for actual applications, including training and entertainment. Figure 2 presents a structural view of Soar[8] in terms of its primitive memories (square-edged modules), processes (round-edged modules), and their connections (arrows). Each of the processing modules can run asynchronously whenever data is available to process.

Starting at the bottom of Figure 1, input comes in through the perception module and is held in the perceptual short-term memory. Symbolic structures are extracted from perceptual short-term memory and deposited in Soar's working memory. Working memory acts as a global short-term memory, and it supports rich relational symbolic structures that can describe complex situations among multiple entities. Typically, this includes current mission information, choices of tactics, and general situational awareness for the robot. The contents of working memory cue the retrieval of knowledge from Soar's three independent long-term memories.

Procedural memory contains Soar's knowledge, encoded as if-then rules, of how to directly select and perform actions, and more details are provided below, in Section 3.1. Soar's semantic memory stores general facts about the world, including facts and properties of objects, declarative descriptions of doctrine and tactics, etc. This is Soar's permanent store of its global world model. As we develop robots that move beyond skill-based tasks such as navigation, to tasks that involve interacting with humans, semantic memory will become more important as a source of a broad range of knowledge. In order to access knowledge relevant to the current situation, Soar's supports associative retrieval of data from semantic memory, and this scales to very large declarative stores. Although some knowledge will build up in this memory as new information is discovered, much of the knowledge can be pre-seeded (and shared across all robots) from existing knowledge bases. Soar also has an episodic memory that stores snapshots of the agent's experiences. This is what the agent remembers and provides an agent with the ability to reflect back on prior experiences and use them to guide future behavior and learning.

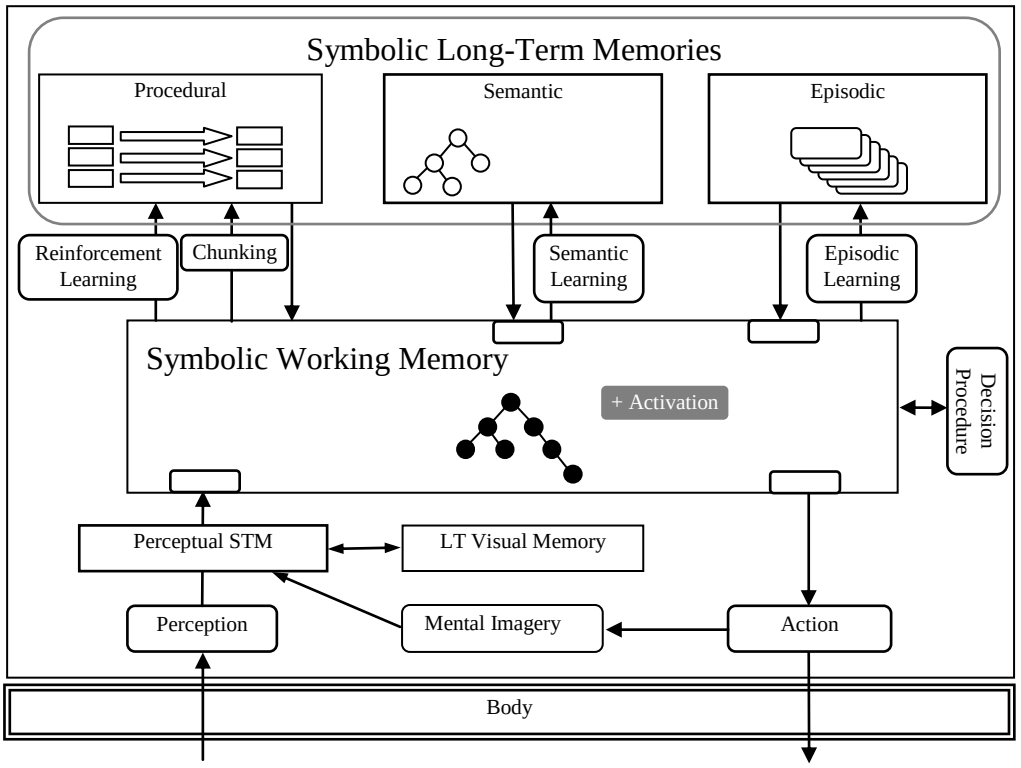

Fig. 2. Soar block diagram.

Robot control architectures invariably make a distinction between their representation of the current situation and their procedural/control knowledge, although some approaches attempt to minimize the independent representation of the current situation[18], much to their own peril. However, many robot architectures ignore semantic and episodic knowledge, or have only limited, task-dependent long-term declarative memories. Agents developed in these approaches are only in the "here and now" and do not have the sense of history made possible by these additional memories.

There are purely functional reasons for multiple memories as well. If we did not decompose an agent's knowledge into short-term and long-term memories, it would be nearly impossible to support efficient reasoning about the current situation, while at the same time supporting extremely large knowledge bases. Although it may appear that a robot needs only limited information about a few tactics, in the real world, a robot will need a wide range of knowledge. Our own experience with TacAir-Soar, a system that could execute U.S. tactical air missions, is that supporting large knowledge bases is critical, and different types of knowledge are needed. Procedural knowledge (shown on the left of Figure 2) directly determines the agent's behavior – it is the knowledge of what to do in a give situation. Semantic knowledge includes facts about the world – what it "knows", while episodic knowledge is an agent's memory of its own experiences – what it "remembers". Soar is designed to support efficient access to each of these memories so that as the system learns it can maintain reactivity. Soar scales to the large bodies of knowledge (including tactics, doctrine, and experience) required in real-world problems (which has yet to be demonstrated by competing cognitive architectures that have been mainly developed with the goal of detailed cognitive modeling, without the co-commitment to functionality in complex, knowledge-rich situations) and can be used in real-time on a wide variety of computer platforms (from iPods to workstations, including Macs, PCs, and Unix systems) and it easily interfaces to software systems, independent of their native language.

3.1 Perception, mental imagery, and motor control

The perceptual short-term memory supports both icon (pixel-level) and diagrammatic (geometric) spatial representations that are built up not only from perception, but also through Soar's mental imagery module (center bottom of figure) that allows the system to overlay previous structures during spatial reasoning and planning. These spatial and visual depictive

representations are critical for representing non-symbolic information, such as terrain, which aids the tactical reasoning facing a military robot[19, 20]. For example, Lathrop developed Soar agents that modeled the perceptual processing of robotic scouts participating in a security and reconnaissance mission[21]. In the scenario, robotic scout vehicles must cooperate to visually track an approaching enemy. In this scenario, a single enemy agent is observed by one vehicle, who communicates it to the other. Mental imagery allows the second agent to create an internal representation of the enemy vehicle, extend it with hypothetical information about possible additional vehicles (a single vehicle never travels alone), and then use non-symbolic processing to imagine possible route for these vehicles through the terrain. Finally, the agent can overlay its mental image with possible positions for itself to determine the best location and orientation for tracking the vehicles as they move through the terrain. This would be difficult to impossible to carry out in a purely symbolic representation, which has led many to incorporate map-based representations in their robotic systems. However, an important extension in our own work is that we have developed task-independent interfaces for interacting with the mental imagery system so that as the uses of mental imagery expand, there is no need to modify or extend its integration within the architecture.

We have also extended the ability to incorporate imagery so that not only can complex actions be initiated through the motor system, but Soar also supports complex actions influencing mental imagery. For example, if the motor system can "imagine" its actions, the agent can simulate its behavior before actually executing it[22, 23].This extends not to just simple translation and rotation, but to arbitrary behavior, such as nonholonomic control of a car. We have used this approach to incorporate modern path planning algorithms, such as RRT, into Soar. The modular design of Soar makes it easy to change the high-level control knowledge (in procedural memory) to support different planning algorithms, while also supporting independent changes to the low-level controllers, to support using those algorithms on vehicles with different control systems.

3.2 Control of Sequential Behavior

Deliberate sequential behavior arises in Soar through context-dependent retrievals from procedural memory of *operators*. Operators are the primitive acts of the system and can either change working memory (internal reasoning), initiate retrieval from other long-term memories (deliberate retrieval), initiate action in the world (motor behavior), or initiate changes to the perceptual short-term memory (mental imagery). It is through operators that deliberate behavior arises. However, operators themselves are not primitive structures. Instead, an operator is decomposed into situation-specific rules for when to propose the operator (the situation is such that an action should be considered), evaluate the operator (how good is the operator in this situation and how does the operator compare to alternatives), and apply the operator (what changes should be made to the current situation – including motor actions). This approach supports flexible, robust behavior, where knowledge to propose an operator can be quite general, whereas evaluation knowledge can be very specific, taking into account perceptual data, and relevant information retrieved from long-term memory. This approach proved to be very successful for encoding not only what behavior is possible in a situation, but also which behavior is appropriate for the current tactic and mission being executed, as well as which behaviors are doctrinally correct. Thus, an agent in Soar may be in exactly the same physical situation at different times, but generate completely different behavior depending on its mission, tactics, and doctrine. This context-dependent behavior is possible in Soar because a combination of specific and general rules can be used for selection and application of operators, in contrast to finite state machines where all possible situations must be enumerated. This process is also more than fast enough to respond to changes in the environment. The basic loop of receiving input, making a decision, and taking an action occurs in ~100 microseconds on standard hardware.

3.2 Hierarchical Behavior

Although, one can think of a mission as consisting of a long series of actions, for most behaviors, it is more appropriate to think of them as a hierarchy of more and more abstract actions. For a simple mission that involves moving from one place to another, the most abstract action could be to execute the mission, the next to go to some location, and below that to go to an intermediate waypoint, which is further decomposed into commands for moving forward and turning. Soar directly supports the encoding and execution of such hierarchical structures. This is accomplished by allowing the selection of abstract operators. Figure 3 shows such a structure for an intercept by a tactical aircraft (taken from TacAir-Soar). When the "intercept" operator is selected, a new goal is automatically generated to apply the operator because it is too complex to implement directly as a primitive motor action. Once a new goal is created, additional operators are proposed for implementing intercept, such as "Achieve proximity to the aircraft", "Search for the aircraft", or "Employ

weapons against the aircraft". The knowledge as to when to select these operators encodes the appropriate doctrine and tactics, which is further refined as an operator is selected and decomposed. When a primitive operator is selected, rules create appropriate commands for the motor system such as "Select a missile to fire", "Push the fire button", and "Set the aircraft to fly a particular heading." This decomposition process is dynamic, in that it depends on the specific situation – so that as the world changes, different operators can be selected at any level of the hierarchy. This supports not just reactive behavior, but reactive behavior modulated by goals.

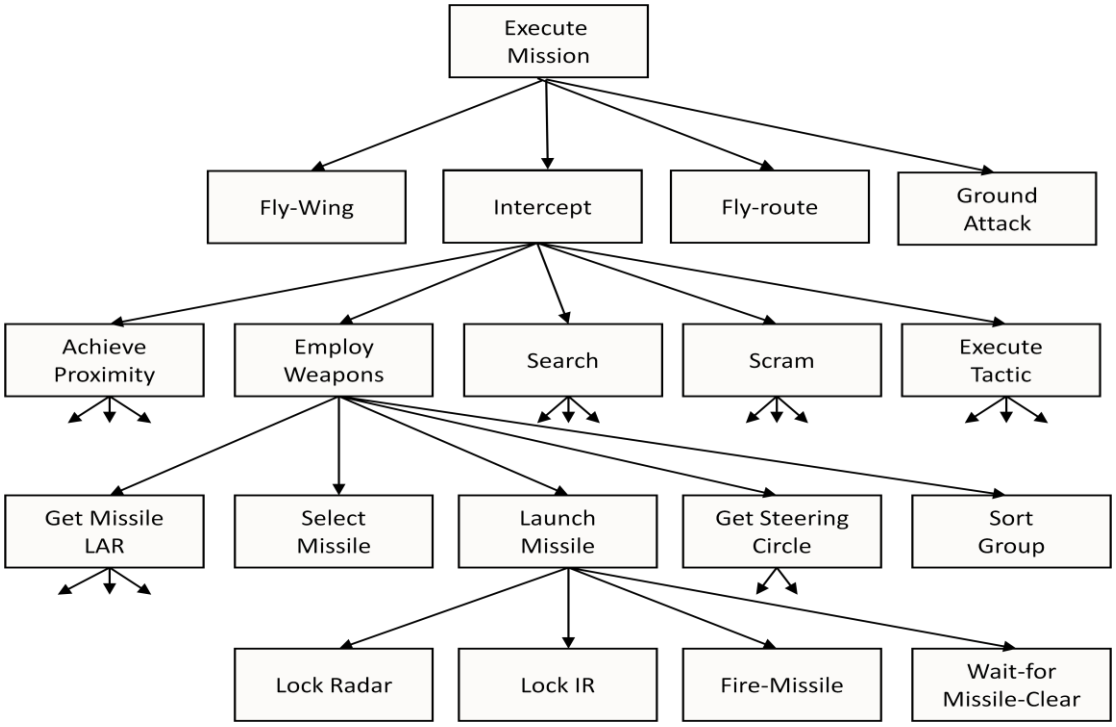

Fig. 3. TacAir-Soar operator hierarchy.

This same approach is used in our physical robot to represent knowledge for executing its missions, although to date the missions and associated behaviors are much simpler. However, based on our previous work with TacAir-Soar, the structure is there to support the extremely complex hierarchical behavior necessary to encode a wide range of doctrine and tactics.

Within this approach, coordination is just another way of pursuing tactical and strategic goals and is part of the knowledge that encodes the appropriate doctrine and tactics. For example, the "scram" action above is selected when another plane is about to take a shot at an enemy and has communicated that the agent might be in harm's way – so the agent must quickly maneuver to get out of the way. More complex coordination, such as flying in formation, is similarly reflected as the fly-wing mission, with communication with other agents being primitive actions that are executed in subgoals.

Although dynamic hierarchical decomposition is important for encoding structured knowledge, such as the goals and subgoals that make up the current mission, there are often multiple goals that need to be pursued that do not map directly onto a hierarchical decomposition. For example, if the robot notices a new entity via perception, it needs to identify it and possibly communicate information about it, while still executing its current mission. To support this behavior, Soar also rapidly switches to a new higher-priority operator, executes it and then returns to the mission execution. Soar is designed so that goal hierarchies such as the one shown in Figure 3 are *re-entrant*, meaning there is sufficient information stored in working memory so that the complete hierarchy will be regenerated if it is ever interpreted. In the limit, Soar can switch between tasks as necessary to respond to the multiple goals it is pursuing.

3.3 Planning

Another issue that arises in trying to create flexible systems is how to handle incomplete or uncertain knowledge, such as when there are multiple paths a robot could take, multiple tactics it could use, or different actions it could employ to execute a tactic. In these cases, the ability to plan ahead and discover the possible interactions and implications of the alternative actions is critical. This occurs in Soar automatically, not just in path or action planning, but across all operators/goals that arise whenever there is uncertainty as to what action to take. In these situations, Soar automatically generates a subgoal in which the Soar agent reasons about the alternates, possibly planning ahead to discover which course of action is best given the current situation and the agent's mission and available knowledge. The planning is not closed off to one type of behavior, but is open so that for example, when the agent is facing a tough decision, it might realize that it can communicate with a human or another robot to aid in making the decision, or if isolated, plan out its behavior on its own. The plans that it develops are conditional on the current situation, so that if things do not play out as expected, the agent will either take an obvious course of action, or it will replan. In addition, the planning and replanning can be sensitive to time so that if ever it is better to take some action than plan, that will also happen.

Planning combines with the hierarchical structures described above, so that the agent can plan at any level of the goal hierarchy. Moreover, if the execution of a level of the hierarchy is routine, internal modeling of the actions required to carry out a goal are skipped. As an example, Figure 4 shows a 2D projection of a plan generated by a Soar agent in the 3D Urban Combat Testbed (UCT), which is a military simulation of small arms combat built on top of Quake 3. On the right of Figure 4 is a depiction of the internal map built up by a Soar agent and the path it traversed in going from the left of the diagram to the right. The large rectangular regions in the middle of the map are buildings that cannot be traversed. To move through the environment, the agent must issue commands that are at the same level of motor commands in a robot: turning and moving forward. However, these commands are suboperators of goals to move from region to region. When planning, executing these primitive operators is avoided by including procedural knowledge that "teleports" the agent from region to region when the move-to-region operator is selected. This greatly speeds planning by concentrating search where there is uncertainty.

Given the dynamics of the world, an agent must be able to handle situations when its plans fail. In Figure 3, the agent first searches its internal map and comes up with a plan to go across the top of the map. However when it reaches region 200, it finds that its path is unexpectedly blocked. At this point, the agent automatically abandons that path because its assumptions about the availability of a path from region 200 to 202 are violated. It once again faces uncertainty and replans. It then follows the new plan, which takes it around the building before it eventually achieves its goal. This same behavior is performed by our Soar robotic agent, although for much smaller areas (because of limits on our laboratory space) and using a graph-based representation of space instead of the region based used in UCT).

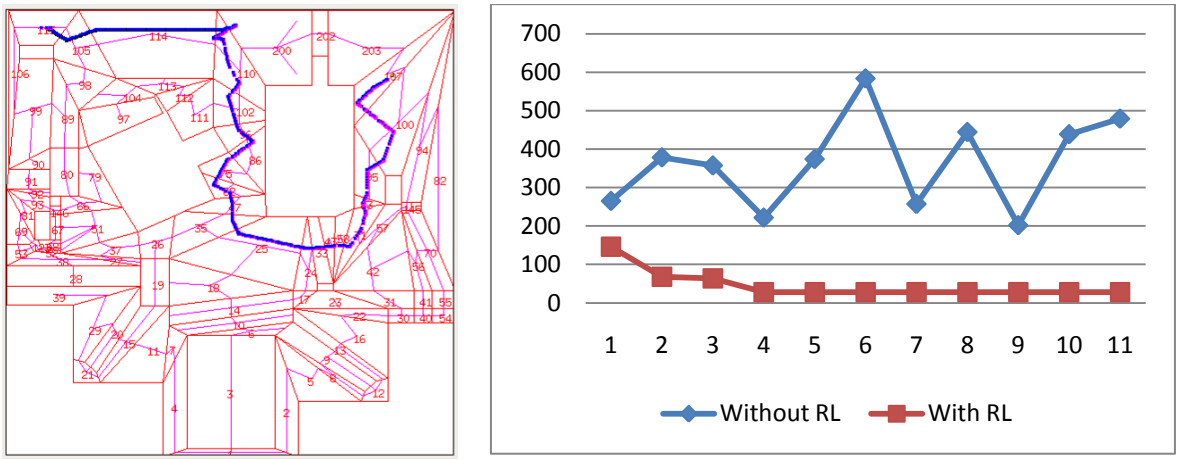

Fig. 4. Results from learning via RL and chunking in Soar (right) in an exploration and navigation task implemented in the Urban Combat Testbed. The values in the graph are the number of areas visited when exploring the map to get from a start location to a goal location.

3.5 Learning: Chunking & Reinforcement Learning

Soar has two learning mechanisms associated with procedural memory: chunking, which compiles the problem solving in subgoals, and reinforcement learning, which tunes the actions of rules that evaluate operators. Chunking continually improves the reactivity of Soar agents by replacing any deliberation in subgoals with reactive selection and execution of operators, while reinforcement learning improves the quality of behavior by tuning the decision making in response to an agent's experience. With chunking, the internal search carried out to find a path is captured as rules, so that in the future, if the same search or a subset is required, rules will directly make the correct decision without search. Moreover, the learned rules are sufficiently specific so that if knowledge about the structure of the environment changes, such as a connection between areas is closed off, the chunks will not match, and replanning will occur. Chunking occurs in parallel with problem solving so that the rules it learns are immediately available.

Reinforcement learning (RL) emphasizes learning functions that associate situations with potential reward based on statistical regularities that arise in an environment in relation to an entity's actions and the rewards it receives from the environment. Within a cognitive architecture, reinforcement learning provides a means for learning control knowledge from experience, specifically statistical correlations related to expected reward. Although analytic learning mechanisms such as Soar's chunking can also learn control knowledge, they require an internal model of the environment, which RL does not. Reinforcement learning has been extensively studied in the last fifteen years, but research has focused on environments where the available tasks, actions, rewards, and features of the environment are known when the agent is constructed. In contrast, our work in RL extends its application to autonomous entities that pursue many different tasks, many whose details are not known at design time. Moreover, by pursing RL in the context of Soar, this research can explore the integration of RL with the other components of Soar. For example, in Soar, RL applies across all goals and subgoals, automatically providing an architecture for hierarchical reinforcement learning[25].

Fig. 4 illustrates an example from an exploration problem in a UCT. In this example, a Soar agent uses RL and chunking to learn control strategies for searching for the goal across multiple trials. The graph shows the number of areas visited by the agent with and without learning. The RL agent quickly learns to outperform an agent without RL. One of the advantages of RL in this situation is that the agent developer did not have to encode the control rules and the agent learned those control policies.

3.5 Learning: Episodic Memory

Episodic memory contains memories of what was experienced in the past. Prior episodes can be recalled to answer questions about the past, to aid in decision making through predicting the outcome of possible courses of action based on the past, possibly by creating an internal model of the entity and its environment based on observations of past outcomes, or to help keep track progress on long-term goals. Important for the work proposed here, the episodic history can also be used for deliberate reflection about past events that can improve behavior through other types of learning, such as reinforcement learning. In Soar, episodic memory includes specific instances of the structures that occur in working memory at the same time, as well as the ability to recall the "next" episode that occurred[26]. This provides the ability to remember the context of past experiences as well as memories of the ongoing experience, which can be used to predict what will happen in similar situations in the future. One advantage of episodic memory is that it is a low-cost, always available learning mechanism that records the complete experience of the robot, independent of the robot's goal. For example, episodic memory saves away memories of the objects that the robot encounters, and if in the future the robot needs some object, such as a tool, it can retrieve from episodic memory where it encountered that tool, and retrieve it.

Although reinforcement learning provides an incremental approach for learning from success and failure, episodic memory provides a more direct path to learn from successes and failures by providing the ability to recall when possible actions either succeeded or failed in similar situations. Figure 5 illustrates the results from a game-world in which simulated tanks fight against one another. In this case, an agent with episodic memory (EM) is pitted against an agent with a hand-coded knowledge-engineered control policy. The figure shows, on average, how the EM agent performs against the hand-coded agent. Initially, the agent lacks knowledge about the specific tactics useful in this domain, and performs poorly. However, using EM and some general knowledge that the agent should pursue actions that result in success and avoids ones that do not, the agent EM learned to beat the fixed-policy controller after about 20 games. The learning gave the agent effective "dodge" and "move and fire" tactics without any explicit user programming.

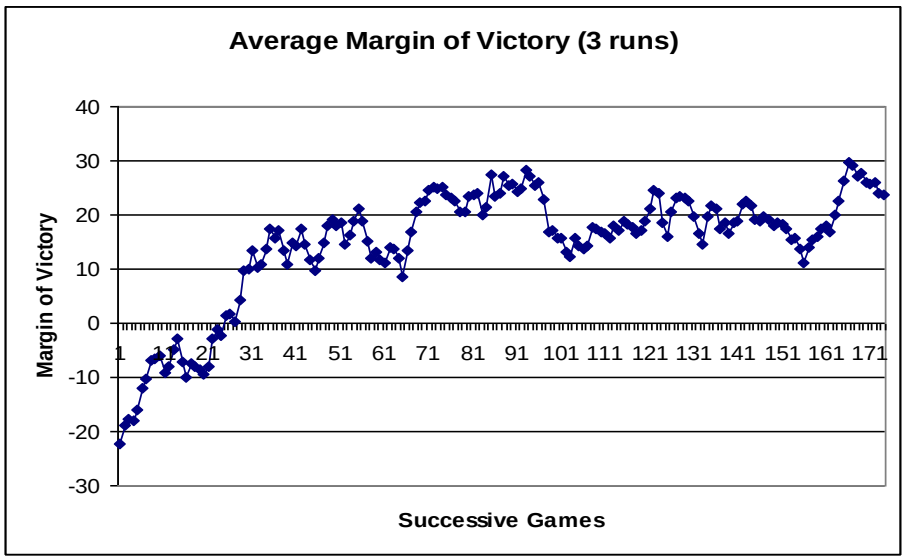

Fig. 5. Agent performance learning from past successes and failures

4. DISCUSSION

The purpose of bringing ideas from cognitive architecture together with robotics systems is to expand the cognitive capabilities of today and tomorrow's unmanned systems. Five years ago, it would have been difficult, if not impossible to predict that we would be using thousands of tele-operated robots for IED disposal. It is probably also impossible to predict what role unmanned systems will play in the military in the next 5-10 years. What we can predict is that the types missions for unmanned systems will explode and change as both the capabilities of unmanned systems improve and as the mission of the military changes. The future demands that our research focus on developing flexible unmanned systems that can learn to perform missions that the original developers never imagined.

To support this, we need to:

1. Develop core cognitive capabilities that will be important no matter how we will use unmanned systems.
2. Develop capabilities that are critical to exploring the space of possible uses. Capabilities that make it easy to experiment with different missions, different goals, different ways of using unmanned systems.
3. Develop capabilities that make it possible for an unmanned system to be quickly retasked to new missions and new uses in the field, and that allow it to adapt quickly to its new missions and role.

Both 2 and 3 will put a premium on many types of learning and instruction, where a human cannot only quickly direct an unmanned system through short commands or gestures, but can communicate complex new behaviors – new missions, new doctrine, and new tactics, through techniques such as learning by demonstration and learning by instruction using language[27, 28]. It will also put a premium on the robot's ability to communicate and explain to a human its understanding of newly acquired knowledge and missions, not just through verbal communications but also through visual presentations and simulations. Moreover, inherent to this approach is that it will be necessary to develop techniques to ensure the correctness of its instructions and its interpretation of them. It will also have to have failsafe mechanisms that ensure that instructions cannot conflict with the robot's core doctrine[29]. These are huge challenges, but need to be pursued.

ACKNOWLEDGMENTS

This research was funded by a grant from US Army TARDEC.

REFERENCES

[1] Langley, P., Laird, J. E., & Rogers, S. "Cognitive architectures: Research issues and challenges," Cognitive Systems Research, 10(2), 141-160 (2009).

[2] Benjamin, P., Lyons, D., and Lonsdale, D., "Designing a Robot Cognitive Architecture with Concurrency and Active Perception," Proceedings of the AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics, October, (2004).

[3] Benjamin, P., Lonsdale, D., and Lyons, D., "Embodying a Cognitive Model in a Mobile Robot," Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, October, (2006).

[4] Trafton, J. G., Schultz, A. C., Cassimatis, N. L., Hiatt, L. M., Perzanowski, D., P., Brock D. P., Bugajska M., & Adams W., "Communicating and Collaborating with Robotic Agents," Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation, edited by R. Sun., 252-278 (2006).

[5] Avery, E., Kelley, T., and Davani, D., "Using Cognitive Architectures to Improve Robot Control: Integrating Production Systems, Semantic Networks, and Sub-Symbolic Processing," Proceedings of the Conference on Behavior Representation in Modeling and Simulation, 15-18 (2006).

[6] Kelley, T.D., "Developing a Psychologically Inspired Cognitive Architecture for Robotic Control: The Symbolic and Subsymbolic Robotic Intelligence Control System (SS-RICS)," International Journal of Advanced Robotic Systems, 3(3), 219-222 (2006).

[7] Long, L.N., Hanford, S.D., Janrathitikarn, O., Sinsley, G.L., and Miller, J.A., "A Review of Intelligent Systems Software for Autonomous Vehicles," Proceedings of the IEEE Symposium Series in Computational Intelligence, (2007).

[8] Albus, J. S., "4D/RCS: A reference model architecture for intelligent unmanned ground vehicles." In Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, (2002).

[9] Laird, J.E., "Extending the Soar Cognitive Architecture," In Proceedings of the First Artificial General Intelligence Conference, (2008).

[10] Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V., "Automated intelligent agents for combat flight simulation." AI Magazine, 20(1), 27-41 (1999).

[11] Benjamin, P., Lonsdale, D., & Lyons, D., "A Cognitive Robotics Approach to Comprehending Human Language and Behaviors. "Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interaction, 185-192, (2007).

[12] Chong, R. S., and Wray, R. E., "Constraints on Architectural Models: Elements of ACT-R, Soar and EPIC in Human Learning and Performance," Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation, and Validation (K. Gluck and R. Pew, eds.), 237-304 (2005).

[13] Wray, R. E., Laird, J. E., Nuxoll, A., Stokes, D., and Kerfoot, A., "Synthetic Adversaries for Urban Combat Training," AI Magazine 26, 82-92 (2005).

[14] Laird, J. E., Hucka, M., Yager, E., & Tuck, C., " Robo-Soar: An integration of external interaction, planning, and learning, using Soar," IEEE Robotics and Autonomous Systems. 8(1-2), 113-129 (1991).

[15] Laird, J. E., & Rosenbloom, P. S., "Integrating execution, planning, and learning in Soar for external environments," Proceedings of the National Conference of Artificial Intelligence, 1022-1029 (1990).

[16] Pearson, D. J., Huffman, S. B., Willis, M. B., Laird, J. E., and Jones, R. M., "A Symbolic Solution to Intelligent Real-Time Control," Robotics and Autonomous Systems 11, 279-291 (1993).

[17] Olson, E., [Robust and Efficient Robotic Mapping], Ph.D. Thesis, MIT, (2008).

[18] Brooks, R. A., [Cambrian Intelligence], MIT Press, (1999).

[19] Lathrop, S. D., and Laird, J. E., "Towards Incorporating Visual Imagery into a Cognitive Architecture," Proceedings of the Eighth International Conference on Cognitive Modeling, Ann Arbor, MI., (2007).

[20] Lathrop, S. D., and Laird, J. E., "Extending Cognitive Architectures with Mental Imagery," Proceedings of the Second Artificial General Intelligence Conference, (2009).

[21] Jaczkowski, J. J., "Robotic technology integration for army ground vehicles," Aerospace and Electronic Systems Magazine, 17, 20-25 (2002).

[22] Wintermute, S. and Laird, J. E., "Bimodal Spatial Reasoning with Continuous Motion," Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, Illinois, (2008).

[23] Wintermute, S., "Integrating Reasoning and Action through Simulation," Proceedings of the Second Conference on Artificial General Intelligence, (2009).

[24] Gorski, N.A. and Laird, J.E., "Investigating Transfer Learning in the Urban Combat Testbed," (Report No. CCA-TR-2007-02). Ann Arbor, MI: Center for Cognitive Architecture, University of Michigan, (2007).
[25] Dietterich, T. G., "An Overview of MAXQ Hierarchical Reinforcement Learning," Choueiry and Walsh (Eds.) Proceedings of the Symposium on Abstraction, Reformulation and Approximation SARA 2000, Lecture Notes in Artificial Intelligence, New York: Springer Verlag, (2000).
[26] Nuxoll, A. M. and Laird, J. E., "Extending Cognitive Architecture with Episodic Memory," Proceedings of the 21st National Conference on Artificial Intelligence, Vancouver, B.C., Canada, (2007).
[27] Allen, J. F., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., & Taysom, W., "PLOW: A collaborative task learning agent," Proceedings of the 21st National Conference on Artificial Intelligence, Vancouver, B.C., Canada, (2007).
[28] Huffman, S. B., & Laird. J. E., "Flexibly Instructable Agents," Journal of Artificial Intelligence Research, 3, 271-324 (1995).
[29] Arkin, R. C., "Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture Part II: Formalization for ethical control," in Proceedings of the First Conference on Artificial General Intelligence, Memphis, TN., (2008).