2010 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
MODELING & SIMULATION, TESTING AND VALIDATION (MSTV) MINI-SYMPOSIUM
AUGUST 17-19 DEARBORN, MICHIGAN

ACHIEVING SEMI-AUTONOMOUS ROBOTIC BEHAVIORS USING THE SOAR COGNITIVE ARCHITECTURE

Robert Marinier, PhD
Robert Bechtel, PhD
Andrew Dallas
Soar Technology, Inc.
Ann Arbor, MI

ABSTRACT

The Soar Cognitive Architecture is a reasoning system that enables knowledge-rich, mission focused reasoning including integration of bottom-up, sensor-driven reasoning and top-down, context-driven reasoning, and more intelligent use of existing sensors. This reasoning is a combination of deliberate (e.g., planning) and reactive (e.g., hard-coded) behaviors. We are applying Soar on a current effort to (1) increase autonomy and (2) achieve equivalent or superior performance while controlling weight, energy, and costs.

INTRODUCTION

Autonomy requires understanding the situation and generating appropriate behaviors. Understanding a complex situation requires the integration of bottom-up (e.g., image processing) and top-down (e.g., contextualized reasoning) processes to achieve satisfactory performance. This integration may include top-down hints to bottom-up processing algorithms, top-down selection of bottom-up processing algorithms, or even top-down control of sensor hardware (e.g., changing shutter speed based on conditions). Similarly, bottom-up processing may present the reasoning system with options (e.g., possible object classifications) that it may select from. Behaving in a complex world requires that autonomous behaviors be robustly and adaptively executed using general context-based reasoning. Robustness requires leveraging broad knowledge of tactics, doctrine, platform capabilities, and the mission to plan and discover novel solutions in the face of unforeseen difficulties and uncertainty.

Moreover, intelligent leveraging of existing sensors and technology can positively impact weight, energy, and cost factors by reducing or eliminating the need to "upgrade" to more expensive sensors. For example, image processing may have a hard time distinguishing between a rock and a bush given a low-resolution camera, but if the reasoner is told it is either a rock or a bush, and it also knows that it is in mountainous terrain with little vegetation, it will conclude that it is probably a rock.

The Soar Cognitive Architecture is a reasoning system that enables these sorts of knowledge-rich, mission focused reasoning. This reasoning is a combination of deliberate (e.g., planning) and reactive (e.g., hard-coded) behaviors. We are applying Soar on a current effort to (1) increase autonomy and (2) achieve equivalent or superior performance while controlling weight, energy, and costs.

THE ROBOT CONTROL STACK

A robot control stack contains multiple levels of control (Figure 1). These system levels provide a means for each level to specialize the kinds of computation it is performing (e.g., regulating voltage across a motor vs. mission-level planning), but also provide a means for interaction between levels.

At the lowest level, there is a robot architecture that interfaces with the hardware, which in turn interfaces with the environment. Most existing non-hardware robotics work is focused on this level. The robot architecture processes raw sensor information for consumption by higher-level processes, and also converts higher-level commands into actuator commands (e.g., translating “move forward at speed X” into smoothly changing voltages across motors). Other algorithms at this level include local obstacle avoidance, and mapping an area. Even higher-level algorithms, like navigation beyond the near-field, are still focused on direct robot control.

Existing interfaces such as Player abstract away the hardware, allowing the same driver software to support interaction between real robots and the real world, simulated robots in simulated worlds, or even a mixture of the two.

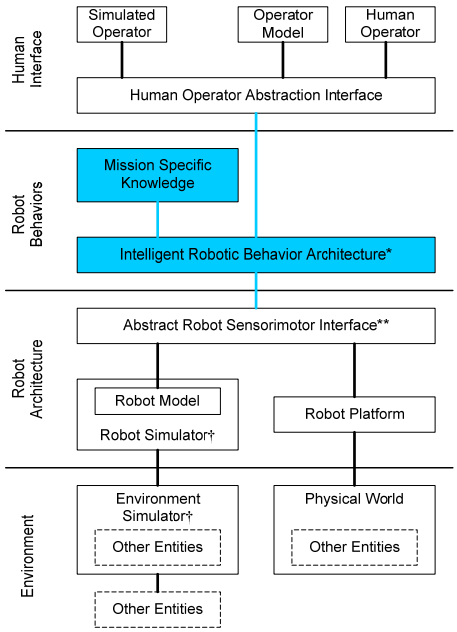

Figure 1: Robot Control Stack

The next level up is the robot behavior level. This is the level that manages the mission that that robot is trying to accomplish. For example, a sophisticated robot architecture may be able to navigate to a specified location, but some higher-level component must still determine which location the robot should go to and under what conditions the robot should go elsewhere. Additionally, the behavior level may be responsible for integrating information from non-organic sensors (e.g., information received about the far-field). These decisions could be reactive (e.g., if performing mission X and sense Y, do Z) or it could be the result of reasoning (e.g., planning). Furthermore, this level can provide top-down context to the robotic architecture level (e.g., the sensors cannot distinguish between a rock and a bush, but the intelligent behavior architecture knows that in this terrain rocks are far more common than bushes, so it is probably a bush). This top-down control can also provide the ability to reconfigure the sensor and motor systems in cases where the combination of the current situation and mission dictate an alternative configuration (e.g., adjusting the shutter speed and zoom of one camera to compensate for another damaged camera in order to best perceive nearby moving objects).

In order to drive these behaviors, the behavior level needs both an intelligent architecture and knowledge specific to the mission it is trying to accomplish. For example, suppose a robot is trying to get supplies to a unit. It encounters a dangerous situation enroute (e.g., terrain it isn't certain it can traverse, or hostile activity). If the supply mission is non-critical, the robot may give up and return to base, or select a new route that is much longer but safer. On the other hand, if the supply mission is critical (e.g., the unit is under attack and running out of ammo) then the robot may decide to take the risk, since that is the only way the supplies can possibly reach the unit on time. The next section of this paper focuses on a possible intelligent robotic behavior architecture.

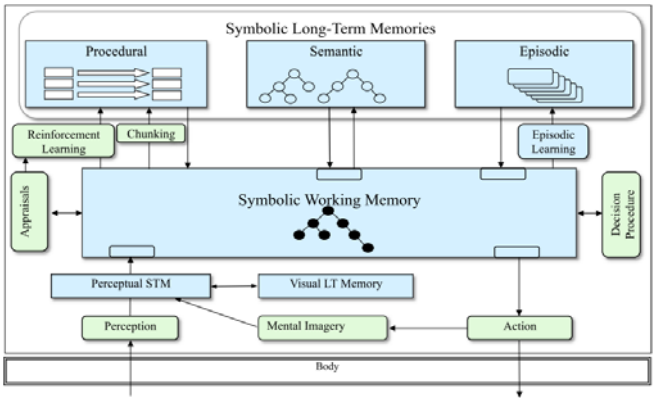

Figure 2: The Soar Cognitive Architecture

The final level is the human interface level. Realistically, the vast majority of robots, even highly autonomous ones, will interact with humans at some level; thus, we describe this approach as applicable to achieving semi-autonomous behaviors, where the exact level of autonomy may vary widely. At one extreme, the interaction is essentially teleoperation, with the human directly controlling the robot. At another extreme, the human merely gives the robot its orders (e.g., via speech), and the robot performs them autonomously. Intermediate levels allow for human-robot teams, where the robot can perform many tasks on its own, but looks to the human for guidance in difficult situations. For example, one vision for robots with weapons is that the robot can perform maneuvering, but a human is required to pull the trigger. At the human interface level, interfaces exist to allow for the possibility of simulated operators or otherwise modeling an operator, primarily for development and testing purposes.

SOAR: AN INTELLIGENT ROBOTIC BEHAVIOR INTERFACE

Soar is a cognitive architecture designed with the goal of achieving human level behavior [1]. Cognitive architectures are different from other agent architectures in that the guiding principle is that complex behavior arises from the interactions of simple, domain- and task-independent components combined with knowledge. For example, while an agent architecture might contain a planning module that performs a specific kind of Partial Order Planning (POP) algorithm, Soar contains general computational mechanisms that can be "programmed" to perform a specific kind of POP planning via the addition of knowledge about how to do that.

One way to view this is that Soar is both? a virtual machine and a programming language. What separates it from other general purpose languages like Java and C++ is that its mechanisms and primitives are designed to support behavior generation, and thus it provides a more useful abstraction of the underlying machine for behavior development.

An advantage of this behavior-centric, knowledge driven approach is that, e.g., a Soar system may have knowledge many planning approaches that are seamlessly applied and even interwoven as the situation changes, thus changing which knowledge is most applicable for the current situation.

While Soar is inspired by human architecture (e.g., the brain) and the ways humans perform various tasks, its focus is on maximizing functionality. This is in contrast to other cognitive architectures (e.g., ACT-R; [2]) that focus on fidelity to human behavior, including timing and errors. Those architectures are primarily focused on understanding human psychology, rather than on advancing artificial intelligence.

Figure 2 shows the current Soar architecture. We will not describe every component in detail, but will touch on a few key aspects.

Soar's working memory is a symbolic graph structure containing a description of the current situation. Other components interact primarily by reading from and writing to working memory. Soar contains perception and action modules that provide a means for external systems to provide information to Soar, and for Soar to provide commands to external systems. This information may be transduced directly from sensors (e.g., location information), or may involve complex processing (e.g., object detection and identification). Actions at this level tend to be at the highest level that the underlying robot architecture can understand (e.g., "go to X").

In order to generate actions in response to perceptions, there are several long-term memories that contain knowledge of what to do in various situations. The knowledge combined with the architecture is called the agent. Procedural memory contains rules of the form "if working memory contains pattern X, then make changes Y to working memory". For example, "if the robot's mission is to go from X to Y to Z, and the robot has reached location Y, set the robot's destination to Z". While the implementation is literally rules, a better way to think about it is that procedural memory is an associative memory – that patterns in working memory trigger the retrieval of knowledge.

Whereas procedural knowledge encodes how to do things, the semantic and episodic memories contain declarative knowledge that describes things. Semantic knowledge encodes facts (e.g., the series of waypoints the robot is supposed to visit, what frequencies to communicate on, etc.), whereas episodic knowledge encodes memories of specific

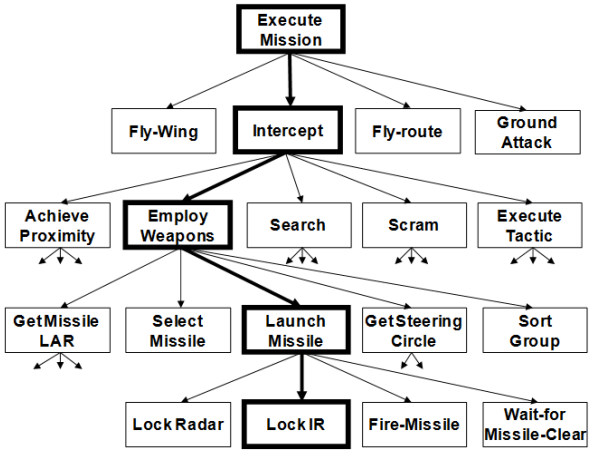

Figure 3: A partial task hierarchy from TacAir-Soar. An example path from mission to behavior is highlighted.

situations (e.g., where the robot was a few minutes ago and what it saw when it was there). One way to think about this distinction is that semantic knowledge encodes what you know, whereas episodic knowledge encodes what you remember. In Soar, retrieving knowledge from semantic or episodic memory is a deliberate action, meaning that the agent's procedural knowledge triggers the retrievals under conditions specified by the procedural knowledge. This retrieved knowledge is added to working memory, which can then trigger additional procedural knowledge (which may perform additional retrievals).

Each memory has associated learning mechanisms. For semantic memory, procedural knowledge can deliberately encode new facts, or update existing facts. For episodic memory, the state of working memory is automatically recorded periodically. Procedural memory actually has two learning mechanisms. One, called chunking, is a way of capturing a long sequence of rule firings in a single rule. Essentially, this captures the results of reasoning, thus avoiding having to repeat similar reasoning in the future. The other is reinforcement learning. In addition to knowledge about how to do things, procedural memory also contains knowledge about how to resolve potential conflicts (e.g., when multiple actions are possible). Reinforcement learning provides a way to tweak this knowledge so that better decisions are made as the agent gains more experience.

The interactions of these mechanisms are controlled by Soar's decision procedure. The decision procedure essentially follows an OODA (Observe, Orient, Decide, Act) loop. First, new perceptions arrive in working memory (Observe). Then knowledge is retrieved from procedural memory that enumerates the various possible actions, including both external actions like moving, and internal

actions like retrieving knowledge from semantic or episodic memory, or manipulating goal structures (Orient). The agent combines these possibilities with preference knowledge specifying which actions are best in which situations to determine which action it should execute next (Decide). The agent then executes that action; for example, sends commands to external systems, performs a retrieval from semantic or episodic, memory, or makes changes to working memory (Act). This loop executes ~20 times per second, providing the ability to quickly react to a dynamic situation. Furthermore, this loop can execute at multiple levels of abstraction. The agent's tasks are typically organized in a hierarchy, and this processing loop is executed to break down high-level tasks (starting with the mission) into lower-level tasks. Figure 3 shows a partial task hierarchy for an agent called TacAir-Soar, which flies simulated fixed-wing aircraft. The highlighted path shows an example transition from mission to behavior, which would have been chosen based on the particulars of the current situation.

GETTING KNOWLEDGE INTO SOAR

The Soar Cognitive Architecture provides a domain- and task-general framework for providing intelligence to a robot. This means that, in order to do anything useful, the various memories have to be loaded with knowledge about the specific domains the robot will operate in and the specific tasks it is to perform. Since Soar is task-independent, it does not impose any requirements on the particular level of abstraction that tasks are specified at – Soar can be used to execute tasks at the mission level (e.g., planning which locations to go to, reasoning about commander's intent, etc.), at the tactical level (e.g., reacting to obstacles in a path, or real-time changes in the environment), or any other level. The difference is in the robotic architecture interface that Soar has to work with, and the knowledge required to take advantage of that interface for the purposes of the task, and the domain knowledge required (e.g., to reason about environment dynamics).

The process for doing this is called KAKE and involves two basic steps: Knowledge Acquisition and Knowledge Engineering. Knowledge acquisition is the process of extracting knowledge about the domain and task from various sources including experts, training materials, SOPs, etc. For example, if we are designing a robot to perform resupply in mountainous terrain under threat from insurgents, we would interview experts who have performed resupply missions under those conditions to try to elicit the various situations that would have to be dealt with, and how those situations are handled. This includes identification of the tasks and subtasks that arise during the mission, the conditions under which they arise, and the actions that should be taken to accomplish those tasks. As in the TacAir-Soar example shown in Figure 3, this should lead to a task hierarchy.

Knowledge engineering is the process of encoding that acquired knowledge into rules and facts that Soar can execute. This may also include the development of some supporting infrastructure for managing the knowledge, although Soar already provides architectural support for much of this, and reusable libraries and tools exist to help develop the rest.

The KAKE process is iterative; once knowledge is acquired and encoded, new questions and issues inevitably arise that require additional knowledge acquisition, and changes to the encoding. In general, developing complex behaviors is a time-consuming process, but tools exist to help streamline this, and others are under development.

AN EXAMPLE

Let's walk through an example. The robot's mission is to bring supplies from a forward operating base to a squad in the field. Essentially, it has to navigate from the base to a rendezvous point, and then return to base. Enroute, the robot sees a person in a red shirt, but as this is irrelevant to the mission, the robot does nothing in response to this observation. Later, the robot receives a message: "New insurgent group active in the area. Distinctive markings include a red shirt." Upon receiving this message, the agent's procedural memory triggers an episodic memory retrieval to see if the robot has encountered these insurgents. The memory of the person in the red shirt is retrieved. The robot executes actions to send a message reporting the sighting, and update the map used by the navigation system to include the insurgent locations. The navigation system plots an alternative route back to the base to avoid this location. Upon arriving at the rendezvous point, the squad loads a seriously injured solider in the vehicle. This person must reach base as soon as possible, but dangerous activity and rough terrain in the area precludes performing an aerial withdrawl. Thus, the agent determines, via the application of procedural knowledge, that the vehicle should take the most direct route back to base, even though it will pass by the insurgent location. The alternative route would mean certain death. Enroute, the vehicle comes under fire, losing one of its camera sensors. The agent adjusts the settings on the remaining camera to compensate, providing a better view than that camera would provide with its default settings. The vehicle makes it back to base, and the solider is saved.

EXISTING SOAR SYSTEMS

The application of Soar to robotic systems is not new. There are several existing Soar systems that demonstrate the approach described above. [3] describes recent thinking in this area

The largest such system is TacAir-Soar [4]. TacAir-Soar flies simulated fixed-wing aircraft; it supports all military missions. While this system is simulated, it contains the same basic interfaces to an underlying platform – inputs with processed sensor data (e.g., "bogey at heading X and distance Y"), and outputs with high-level commands (e.g., "turn to heading X").

Another relevant system is ECGF, which provides a consistent interface between high-level commands and low-level execution. In our description of the robot stack above, we drew the line between the behaviors and the underlying robot architecture in a particular place, but the reality is that this is a gray area. For example, some robotic architectures may support navigation in the far field, while others may not. ECGF [5] exposes an interface that makes it look like all underlying systems support navigation in the far-field (among other behaviors), and implements this navigation for those systems that do not actually support it using their available primitives. This makes higher-level reasoning more portable across platforms.

Soar has also been connected to real robots. Early work on such systems included Robo-Soar and Hero-Soar [6]. These systems performed simple tasks involving manipulating blocks, but demonstrated key aspects of robotic control, including reactivity, planning, and learning within the Soar architecture. Currently, SoarTech is involved in the SUMET program under ONR, which aims to have robotic vehicles perform militarily relevant missions such as resupplying units in the field. Additionally, SoarTech's Robotic Wingman paper at this symposium [7] describes the application of Soar to a robotic system intended to enhance the effectiveness of combat platoons.

CONCLUSION

We have described a robot control stack that includes interaction between low-level hardware control and high-level mission control. We argued that interaction between these levels is critical in taking the next step in autonomous, affordable (in terms of energy, size, and cost) robotics, since more intelligent high-level control will result in more effective use of low-level capabilities. We described the Soar cognitive architecture as a system capable of fulfilling the role of high-level mission control. Finally, we described existing systems that take steps in this direction.

REFERENCES

[1] J. Laird, "Extending the Soar Cognitive Architecture", Proceedings of the First Conference on Artificial General Intelligence, 2008.

[2] J. Anderson, "How Can the Human Mind Exist in the Physical Universe?", Oxford University Press, New York, 2007.

[3] J. Laird, "Toward Cognitive Robotics", SPIE Defense and Sensing Conferences, 2009.

[4] R. Jones, J. Laird, P. Nielsen, K. Coulter, P. Kenny, F. Koss, "Automated Intelligent Pilots for Combat Flight Simulation", AI Magazine, 20(1), 1999.

[5] B. Stensrud, G. Taylor, B. Schricker, J. Montefusco, J. Maddox. "An Intelligent User Interface for Enhancing Computer Generated Forces", Proceedings of the 2008 Fall Simulation Interoperability Workshop, 2008.

[6] J. Laird, P and Rosenbloom, "Integrating Execution, Planning, and Learning in Soar for External Environments", AAAI-90 Proceedings, 1990.

[7] J. Lane, F. Antenori, and A. Dallas. "Robotic Wingman", 2010 NDIA Ground Vehicle Systems Engineering and Technology Symposium, 2010.