Proposal Abstract for DARPA IPTO

BAA #02-21: Cognitive Information Processing Technology

A Novel Cognitive Architecture for Intelligent Systems

Technical contact:

Dr. Pat Langley
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306
Phone: (650) 494-3884
Fax: (650) 494-1588
Email: LANGLEY@ISLE.ORG

Administrative contact:

Dr. Daniel Shapiro
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306
Phone: (831) 234-3098
Fax: (650) 494-1588
Email: DGS@ISLE.ORG

Summary of costs:

Total: $2,500,000
Year 1: $450,000
Year 2: $475,000
Year 3: $500,000
Year 4: $525,000
Year 5: $550,000

Type of business: Other nonprofit

Innovative Claims

We will develop a novel cognitive architecture – Icarus – that combines and extends previously disconnected ideas in a unified manner, and we will demonstrate the capabilities this integration makes possible. Innovative claims for the architecture include:

1. Icarus supports cognition, perception, and action in a single framework. Most cognitive architectures have grown out of research on problem solving and, when they address perception and action at all, these have been added in hindsight. Most reactive architectures focus on perception and action but ignore cognitive activities like problem solving and reasoning. In contrast, our framework gives equal importance to these three facets of intelligent behavior.
2. Icarus supports long-term memories for both concepts and skills. Most cognitive architectures give center stage to knowledge about skills, cast as production rules, operators, or plans, and ignore the role that concepts and categorization play in cognition. In contrast, our architecture organizes categories in a conceptual memory and recognizes concept instances by sorting them through this memory, which it also uses to index and retrieve skills.
3. Icarus combines the logical organization of complex skills with decision-theoretic utility functions. Most cognitive architectures represent both the structure of skills and preferences among them in symbolic form, whereas some reactive frameworks encode utilities but provide little structure. In contrast, Icarus organizes skills in a hierarchical manner, but associates with each one a learned utility function that responds dynamically to environmental cues.
4. Icarus interleaves deliberative planning with reactive execution to achieve flexible behavior in novel situations and efficient behavior in familiar ones. Most cognitive architectures emphasize deliberative problem solving, whereas reactive architectures typically rely on prestored control knowledge or implausibly slow learning. In contrast, our framework executes stored skills in a reactive manner when they are available and uses planning, guided by utility functions, to generate and store new skills when needed.
5. Icarus selects its own top-level tasks, pursues them over time, and abandons them when appropriate. Cognitive architectures usually assume a human will provide a top-level task or goal, then continue to work on it until succeeding or exhausting all options. Most reactive architectures are fickle and have little notion of persistence on a task. In contrast, Icarus selects new top-level tasks with high estimated utilities, uses observed results to revise these estimates, and abandons tasks when it makes insufficient progress on them.

We predict that these innovative features will provide Icarus with more flexible, adaptive, and robust behavior in complex, dynamic domains that involve cognition, perception, and action, and we will test these predictions by developing Icarus knowledge bases for a number of such domains and observing its behavior.

The proposed research will contribute to the fields of knowledge representation and memory, planning and reasoning, execution and control, and machine learning. More important, it will reveal ways in which these distinct disciplines must interact to support generalized cognitive behavior.

Motivation and Background

We propose to design, implement, and evaluate a novel cognitive architecture that integrates perception, action, and cognition to support complex, adaptive behavior in intelligent agents. Our research will make basic contributions to the representation, utilization, and acquisition of knowledge that supports such activities, and we will demonstrate these functionalities on a number of challenging tasks.

The cognitive architecture will enable behavior like that needed on a reconnaissance mission flown by an unmanned air vehicle. Here an autonomous physical agent must decide on its own tasks, manage its cognitive and physical resources, recognize familiar situations and respond to them efficiently, develop its own plans for novel situations, pursue its mission objectives while avoiding dangers, and report back its findings at regular intervals. However, the architecture will be general, and we intend to evaluate it in a variety of domains.

Our technical approach builds on Icarus (Shapiro et al., 2001), an existing architecture which represents knowledge about skills that involve sensing the environment and reacting accordingly. Each skill specifies an objective, the means to achieve it, and a set of requirements. Both objectives and requirements are conjunctions of first-order literals. Icarus organizes these skills in hierarchical manner, with each literal referring to another skill, a primitive action, or a sensor. Each skill has an associated utility function that calculates the expected value of executing that skill, on the current cycle, from values of sensory attributes.

Like other cognitive architectures, Icarus operates in cycles. On each cycle, the interpreter first checks the objective of the top-level skill. If this is satisfied, the system does nothing; otherwise, it examines the requires field to see if the preconditions for action are met. If a nonprimitive requirement is unsatisfied, Icarus calls recursively on the associated skill in an effort to satisfy it. If all requirements are met, the interpreter calculates the expected value for each alternative means and selects the one with the highest utility. The system continues this process until it reaches a primitive action or sensor call, which it then executes. On the next cycle, the architecture takes the new environmental state into account in its decisions. Icarus learns the expected values for skills using a hierarchical variant of SARSA, a model-free method that propagates delayed rewards backward through time.

We have used Icarus to encode skills for a number of tasks, including driving an automobile on a simulated freeway. The architecture's strengths include its hierarchical organization of reactive skills, its clean separation between the logical structure of these skills and their utilities, and its use of skill knowledge to greatly speed the process of learning utilities from delayed reward. Weaknesses include an emphasis on skills rather than other forms of knowledge, a focus on model-free reactive control with no ability to plan or schedule, and limited abilities for short-term memory, attention, or belief maintenance.

We propose to remedy these weaknesses in our research program. As we describe below, we will extend Icarus to incorporate a long-term conceptual memory that supports categorization, introduce action models that allows mechanisms for prediction, plan generation, and resource allocation, and provide a short-term memory that supports focus of attention, belief maintenance, and task generation, persistance, and abandonment.

Conceptual Memory and Category Recognition

The current implementation of Icarus includes a long-term memory that organizes skills in terms of their component subskills, but it has no notion of concepts or categories, which are equally central to cognition. We maintain that concepts are more primary than skills or plans, and that the latter should be cast in terms of concepts. We propose to extend Icarus

to support concepts defined as logical conjunctions of literals that are organized in an is-a hierarchy, much as in existing description logics. In this framework, the requires field for each skill corresponds to such a logical concept, indexing skills in the concept hierarchy in the same way that production systems index their rules with Rete networks. This conceptual organization on memory complements the activity-oriented organization provided by skills.

Recognizing instances of concepts in long-term memory occurs by sorting primitive literals, which correspond to sensory input, down through the concept hierarchy, with arguments being unified along the way. This process happens on every cycle, being driven by new primitive literals that have been added to short-term memory (discussed below) through perceptual mechanisms. The overall mechanism is akin to sorting items through a discrimination network, but the ability to handle relations among literals makes it closer to matching in a Rete network. Each recognized instance (i.e., conjunction of literals) is stored with the concept that matches it. This 'bottom-up' or data-driven retrieval mechanism complements the 'top-down' or intention-driven method already used for action selection. Both are needed for an architecture that must pursue goals while remaining alert for new opportunities.

For this effort, we will not support unsupervised learning of new concepts from observations. Our previous experience with concept formation (e.g., Iba & Langley, 2001) suggests this is a difficult problem, especially when other modules depend on a concept hierarchy whose structure is changing over time. However, we can automate construction of the concept hierarchy from user-specified concepts, including the indexing of skills. The basic method is the similar to that used to incorporate knowledge into description logics by sorting new concepts to their appropriate location. Thus, Icarus will support the incremental introduction of new concepts and skills, but not the autonomous creation of new concepts. Later versions will associate probabilities with literals that are learned ftom experience, with the recognition process producing a degree of belief for each concept instance.

Prediction, Planning, and Resource Allocation

The current version of Icarus has a model of agent intent but no explicit knowledge about the effects of actions on the environment. We plan to extend the architecture's representation of knowledge to encode the expected results of executing actions and skills. In some sense, the objectives field already describes expected outcomes, but this gives no way to measure progress towards a goal. In response, we will incorporate ideas from qualitative physics, associating with each action and skill one or more qualitative changes that its execution should produce. More advanced versions of the architecture will include probabilistic models that describe uncertain effects, along with quantitative models that predict rates of change.

The extended architecture can utilize action models of this sort to predict a skill's effects and determine whether intended results are occurring. Icarus will also use these models for extended projections about the expected effects of skill sequences. As we discuss below, violated expectations should, in the short term, modulate calculations about the values of alternative actions. In the longer term, trace data about the effects of actions and skills will be used to update models through learning. Initial versions will employ simple statistical methods to induce the qualitative changes associated with an action or skill, whereas later versions will use equation discovery methods to induce numeric models that predict effects as a function of sensory variables.

As it stands, Icarus can execute prestored plans or skills, but the addition of predictive models will also let it generate new reactive skills through planning. This process will incorporate standard methods for backward chaining from known objectives to select known skills that achieve them, checking to see if requirements are already met, and recursing if

not. In this scheme, existing skills play the role of macro-operators that hide details about how to achieve an objective. The result is a hierarchical set of skills that, if executed, should transform the current environment into a desired one, as in traditional planners. Additional effort can produce conditional reactive plans that follow different paths to the same end.

The ability to generate plans raises the issue of how planning relates to the execution of existing skills. We assume that plan generation and skill execution utilize different cognitive resources, so that they can occur in parallel. However, we want plans about future action to influence the selection of current actions. One mechanism is to modulate the expected value of executing the current action using the estimated value of the current plan, much as the existing learning method propagates value backwards over a sequence of actions. Icarus' current method for acquiring utility functions can also be used to improve planning, in that the system can learn which plans to expand, given predicted world states rather than observed ones. This should mitigate the utility problem – the slowing down of behavior as one adds knowledge to memory – that has plagued many existing architectures.

Another limitation of the current architecture is its restriction to executing one skill on each time step. Future versions will introduce the ability to execute actions in parallel, but place resource constraints on this ability. This will require an expanded formalism for skills that specifies the resources they consume on each cycle. We will also need to generalize Icarus' current method for skill selection to support resource allocation and scheduling in both reactive execution of skills and in projective planning. Such decisions will be handled at the architectural level, so that the basic scheduling method will remain constant. However, the system should be able to change the specific schedules it constructs by learning estimates for the resources consumed when executing particular skills.

Short-Term Memory, Perceptual Attention, and Belief Maintenance

The introduction of mechanisms for concept recognition and plan generation indicate the need for a short-term memory that holds current beliefs and intentions. We can view this as the active portion of long-term memory, which means that every literal in the short-term store must refer to an instance of some concept or skill in long-term memory. Each such literal has three associated numbers to encode different types of information – one indicating intent (the expected value for a skill or concept), another measuring belief (the probability that a concept is present), and a third for expectations (the probability that a concept will occur in a future state). The initial version will assign one or zero to beliefs and expectations, whereas later versions will assign probabilities. Later incarnations will include time stamps on short-term memory elements that will a support simple version of episodic memory.

The current version of Icarus senses its environment on every cycle, but this happens automatically whenever literals in the requires field must be tested. We intend to place this activity under cognitive control by making sensing actions part of the execution and planning processes. Low-level sensing will still take place automatically, but moving information from this preconscious level into short-term memory will require action on the system's part. Such sensing actions and skills have their own expected values and resource demands, so the above mechanisms for skill selection support naturally ideas of attention and selective sensing.

The existing architecture also supports persistence of primitive literals over time when they have not been sensed recently. However, we must extend this persistence to higher-level concepts that Icarus recognizes as a result of its perceptions. Addition to short-term memory of new beliefs and expectations about the environment will be handled by the recognition process described earlier, but this is not sufficient. We will also need a belief-maintenance process to remove beliefs and expectations (or change their time stamps) when

perceptual information ceases to support them. To this end, we plan to index each literal in short-term memory with the primitive literals on which it depends, and to remove the former whenever the latter become false. This in turn will influence the planning and execution process, since skills are indexed by their requires fields. Later versions of the architecture will extend this technique to update probabilities associated with beliefs and expectations.

Most cognitive architectures assume that tasks are set by an external agent, and the system pursues them until they have been accomplished or it has exhausted all possible approaches. In contrast, humans determine their own high-level tasks, but often give up and move on to others when they are not making sufficient progress. The current version of Icarus, being fully reactive, has no trouble shifting from one skill (which correspond to tasks or subtasks) to another with higher value. But it must be given a top-level task, and it has no notion of persistence, which in some cases can produce oscillation among the selected skills. We plan to extend Icarus to utilize the concept recognition process outlined above to nominate skills with matching requires fields and high expected value, thus letting the system set its own top-level tasks. Moreover, the new version will achieve persistence through an inertia term that modulates expected values and slows their rate of change. Icarus will also modulate expected values based on differences between expected and observed results, abandoning top-level tasks whose revised values fall below those for other nominated tasks.

Distinctive Features and Evaluation Plans

As discussed earlier, our framework integrates features from previous research that have not been combined in a unified cognitive architecture. These include: equal treatment of cognition, perception, and action; interconnected long-term memories for both concepts and skills; merging the logical structure of skills with decision-theoretical utilities; interleaving flexible deliberative planning with efficient reactive execution; and the ability to select top-level tasks, pursue them over extended periods, but abandon them when appropriate. Existing cognitive architectures like Soar and ACT handle some capabilities but not others, whereas reactive architectures typically support their complement. In contrast, Icarus unifies ideas from both traditions.

Our initial efforts will extend the current architecture to include a conceptual memory with retrieval mechanisms, models of action effects used by a plan generation module, and a short-term memory for concepts and intentions. The next stage will expand Icarus' abilities to maintain beliefs, learn action models from experience, allocate resources, and support nomination and abandonment of tasks. The third stage will extend the framework to support probabilistic concepts and quantitative action models, including their revision over time, and selective attention. We plan to demonstrate the new capabilities made possible by these architectural features in a variety of settings, thus providing evidence for the generality of both the architecture and its components. Our aim will be to show that Icarus supports new capabilities not handled by existing architectures.

Because Icarus emphasizes embodied agents that not only think but also sense and act, most testbeds should require such abilities. These will include: flying a simulated unmanned air vehicle on an extended reconnaissance mission to obtain intelligence and report back to headquarters; driving an automobile from one location to another in the presence of signals, obstacles, and other vehicles; solving difficult physics problems that involve diagrams; controlling a mobile robot in an office setting that interacts with humans, avoid obstacles, and seeks out novel situations; and answering textual questions posed by a human about its activities on these tasks. This work will take advantage of simulators and component software developed in our previous work (Shapiro et al., 2001; Yamauchi et al., 1998).