2000

Learning in "Conscious" Software Agents

S. Franklin

Learning in "Conscious" Software Agents

Stan Franklin[1, 2]

Institute for Intelligent Systems and
Department of Mathematical Sciences
The University of Memphis
stan.franklin@memphis.edu
www.msci.memphis.edu/~franklin

Abstract

Here we describe mechanisms for a half-dozen or so different types of learning to be implemented in "conscious" software agents, and speculate briefly on their implications for human learning and development. In particular, we're concerned with the role of attention, that is, bringing specific content to consciousness, in learning. We offer computational mechanisms for such learning. We believe that the "conscious" software agent technology described below, when it matures, will allow the automation of all sorts of human information and decision making agents.

Autonomous Agents

Artificial intelligence pursues the twin goals of understanding human intelligence and of producing intelligent software and/or artifacts. Designing, implementing and experimenting with autonomous agents furthers both these goals in a synergistic way. An *autonomous agent* (Franklin & Graesser 1997) is a system situated in, and part of, an environment, which senses that environment, and acts on it, over time, in pursuit of its own agenda. In biological agents, this agenda arises from evolved in drives and their associated goals; in artificial agents from drives and goals built in by its creator. Such drives, which act as motive generators (Sloman 1987), must be present, whether explicitly represented, or expressed causally. The agent also acts in such a way as to possibly influence what it senses at a later time. In other words, it is structurally coupled to its environment (Maturana 1975, Maturana et al. 1980). Biological examples of autonomous agents include humans and most animals. Non-biological examples include some mobile robots, and various computational agents, including artificial life agents, software agents and many computer viruses. We'll be concerned with autonomous software agents, designed for specific tasks, and 'living' in real world computing systems such as operating systems, databases, or networks.

Global Workspace Theory

The material in this section is from Baars' two books (1988, 1997) (1988, 1997) and superficially describes his global workspace theory of consciousness.

In his global workspace theory, Baars, along with many others (e.g. (Minsky 1985, Ornstein 1986, Edelman

[1]Supported in part by NSF grant SBR-9720314 and by ONR grant N00014-98-1-0332

[2]With essential contributions from the Conscious Software Research Group including Art Graesser, Satish Ambati, Ashraf Anwar, Myles Bogner, Derek Harter, Arpad Kelemen, Irina Makkaveeva, Lee McCauley, Aregahegn Negatu, Fergus Nolan, Hongjun Song, Uma Ramamurthy, Zhaohua Zhang

1987)) , postulates that human cognition is implemented by a multitude of relatively small, special purpose processes, almost always unconscious. (It's a multiagent system.) Communication between them is rare and over a narrow bandwidth. Coalitions of such processes find their way into a global workspace (and into consciousness). This limited capacity workspace serves to broadcast the message of the coalition to all the unconscious processors, in order to recruit other processors to join in handling the current novel situation, or in solving the current problem. Thus consciousness in this theory allows us to deal with novelty or problematic situations that can't be dealt with efficiently, or at all, by habituated unconscious processes. In particular, it provides access to appropriately useful resources, thereby solving the relevance problem.

All this takes place under the auspices of contexts: goal contexts, perceptual contexts, conceptual contexts, and/or cultural contexts. Baars uses goal hierarchies, dominant goal contexts, a dominant goal hierarchy, dominant context hierarchies, and lower level context hierarchies. Each context is, itself a coalition of processes. Though contexts are typically unconscious, they strongly influence conscious processes.

Baars postulates that learning results simply from conscious attention, that is, that consciousness is sufficient for learning. There's much more to the theory, including attention, action selection, emotion, voluntary action, metacognition and a sense of self. I think of it as a high level theory of cognition.

"Conscious" Software Agents

A "conscious" software agent is defined to be an autonomous software agent that implements global workspace theory. (No claim of sentience is being made.) I believe that conscious software agents have the potential to play a synergistic role in both cognitive theory and intelligent software. Minds can be viewed as control structures for autonomous agents (Franklin 1995). A theory of mind constrains the design of a "conscious" agent that implements that theory. While a theory is typically abstract and only broadly sketches an architecture, an implemented computational design provides a fully articulated architecture and a complete set of mechanisms. This architecture and set of mechanisms provides a richer, more concrete, and more decisive theory. Moreover, every design decision taken during an implementation furnishes a hypothesis about how human minds work. These hypotheses may motivate experiments with humans and other forms of empirical tests. Conversely, the results of such experiments motivate corresponding modifications of the architecture and mechanisms of the cognitive agent. In this way, the concepts and methodologies of cognitive science and of computer science will work synergistically to enhance our understanding of mechanisms of mind (Franklin 1997).

"Conscious" Mattie

"Conscious" Mattie (CMattie) is a "conscious" clerical software agent (McCauley & Franklin 1998, Ramamurthy et al. 1998, Zhang et al. 1998, Bogner et al. in press). She composes and emails out weekly seminar announcements, having communicated by email with seminar organizers and announcement recipients in natural language. She maintains her mailing list, reminds organizers who are late with their information, and warns of space and time conflicts. There is no human involvement other than these email messages. CMattie's cognitive modules include perception, learning, action selection, associative memory, "consciousness," emotion and metacognition. Her emotions influence her action selection. Her mechanisms include variants and/or extensions of Maes' behavior nets (1990) , Hofstadter and Mitchell's Copycat architecture (1994) , Jackson's pandemonium theory (1987), Kanerva's sparse distributed memory (1988) , and Holland's classifier systems (Holland 1986) .

IDA

IDA (Intelligent Distribution Agent) is a "conscious" software agent being developed for the US Navy (Franklin et al. 1998) . At the end of each sailor's tour of duty, he or she is assigned to a new billet. This assignment process is called distribution. The Navy employs some 200 people, called detailers, full time to effect these new assignments. IDA's task is to

facilitate this process, by playing the role of detailer. Designing IDA presents both communication problems, and action selection problems involving constraint satisfaction. She must communicate with sailors via email and in natural language, understanding the content and producing life-like responses. Sometimes she will initiate conversations. She must access a number of databases, again understanding the content. She must see that the Navy's needs are satisfied, for example, the required number of sonar technicians on a destroyer with the required types of training. In doing so she must adhere to some ninety policies. She must hold down moving costs. And, she must cater to the needs and desires of the sailor as well as is possible. This includes negotiating with the sailor via an email correspondence in natural language. Finally, she must write the orders and start them on the way to the sailor. IDA's architecture and mechanisms are largely modeled after those of CMattie, though more complex. In particular, IDA will require improvised language generation where for CMattie scripted language generation sufficed. Also IDA will need deliberative reasoning in the service of action selection, where CMattie was able to do without. Her emotions will be involved in both of these.

"Conscious" Software Architecture and Mechanisms

In both the CMattie and IDA architectures the processors postulated by global workspace theory are implemented by codelets, small pieces of code. These are specialized for some simple task and often play the role of demon waiting for appropriate condition under which to act. The apparatus for producing "consciousness" consists of a coalition manager, a spotlight controller, a broadcast manager, and a collection of attention codelets who recognize novel or problematic situations (Bogner 1999, Bogner et al. in press). Each attention codelet keeps a watchful eye out for some particular situation to occur that might call for "conscious" intervention. Upon encountering such a situation, the appropriate attention codelet will be associated with the small number of codelets that carry the information describing the situation. This association should lead to the collection of this small number of codelets, together with the attention codelet that collected them, becoming a coalition. Codelets also have activations. The attention codelet increases its activation in order that the coalition might compete for "consciousness" if one is formed.

In CMattie and IDA the coalition manager is responsible for forming and tracking coalitions of codelets. Such coalitions are initiated on the basis of the mutual associations between the member codelets. At any given time, one of these coalitions finds it way to "consciousness," chosen by the spotlight controller, who picks the coalition with the highest average activation among its member codelets. Global workspace theory calls for the contents of "consciousness" to be broadcast to each of the codelets. The broadcast manager accomplishes this.

Both CMattie and IDA depend on a behavior net (Maes 1990) for high-level action selection in the service of built-in drives. Each has several distinct drives operating in parallel. These drives vary in urgency as time passes and the environment changes. Behaviors are typically mid-level actions, many depending on several codelets for their execution. A behavior net is composed of behaviors and their various links. A behavior looks very much like a production rule, having preconditions as well as additions and deletions. A behavior is distinguished from a production rule by the presence of an activation, a number indicating some kind of strength level. Each behavior occupies a node in a digraph (directed graph). The three types of links of the digraph are completely determined by the behaviors. If a behavior X will add a proposition b, which is on behavior Y's precondition list, then put a successor link from X to Y. There may be several such propositions resulting in several links between the same nodes. Next, whenever you put in a successor going one way, put a predecessor link going the other. Finally, suppose you have a proposition m on behavior Y's delete list that is also a precondition for behavior X. In such a case, draw a conflictor link from X to Y, which is to be inhibitory rather than excitatory.

As in connectionist models, this digraph spreads activation. The activation comes from activation stored in the behaviors themselves, from the environment, from drives, and from internal states.

The environment awards activation to a behavior for each of its true preconditions. The more relevant it is to the current situation, the more activation it's going to receive from the environment. This source of activation tends to make the system opportunistic. Each drive awards activation to every behavior that, by being active, will satisfy that drive. This source of activation tends to make the system goal directed. Certain internal states of the agent can also send activation to the behavior net. This activation, for example, might come from a coalition of codelets responding to a "conscious" broadcast. Finally, activation spreads from behavior to behavior along links. Along successor links, one behavior strengthens those behaviors whose preconditions it can help fulfill by sending them activation. Along predecessor links, one behavior strengthens any other behavior whose add list fulfills one of its own preconditions. A behavior sends inhibition along a conflictor link to any other behavior that can delete one of its true preconditions, thereby weakening it. Every conflictor link is inhibitory. Call a behavior *executable* if all of its preconditions are satisfied. To be acted upon a behavior must be executable, must have activation over threshold, and must have the highest such activation. Behavior nets produce flexible, tunable action selection for these agents.

Action selection via behavior net suffices for CMattie due to her relatively constrained domain. IDA's domain is much more complex, and requires deliberation in the sense of creating possible scenarios, partial plans of actions, and choosing between them. For example, suppose IDA is considering a sailor and several possible jobs, all seemingly suitable. She must construct a scenario for each of these possible billets. In each scenario the sailor leaves his or her current position during a certain time interval, spends a specified length of time on leave, possibly reports to a training facility on a certain date, and arrives at the new billet with in a given time frame. Such scenarios are valued on how well they fit the temporal constraints and on moving and training costs.

Scenarios are composed of scenes. IDA's scenes are organized around events. Each scene may require objects, actors, concepts, relations, and schema represented by frames. They are constructed in a computational workspace corresponding to working memory in humans. We use Barsalou's perceptual symbol systems as a guide (1999) . The perceptual/conceptual knowledge base of this agent takes the form of a semantic net with activation called the slipnet. The name is taken from the Copycat architecture that employs a similar construct (Hofstadter et al. 1994). Nodes of the slipnet constitute the agent's perceptual symbols. Pieces of the slipnet containing nodes and links, together with codelets whose task it is to copy the piece to working memory constitute Barsalou's perceptual symbol simulators. These perceptual symbols are used to construct scenes in working memory. The scenes are strung together to form scenarios. The work is done by deliberation codelets. Evaluation of scenarios is also done by codelets.

Deliberation, as in humans, is mediated by the "consciousness" mechanism. Imagine IDA in the context of a behavior stream whose goal is to find a billet for a particular sailor. Perhaps a behavior executes to read appropriate items from the sailor's personnel database record. Then, possibly, comes a behavior to locate the currently available billets. Next might be a behavior that runs each billet and that sailor through IDA's constraint satisfaction module, producing a small number of candidate billets. Finally a deliberation behavior may be executed that sends deliberation codelets to working memory together with codelets carrying billet information. A particular billet's codelets wins its way into "consciousness." Scenario building codelets respond to the broadcast and begin creating scenes. This scenario building process, again as in humans, has both it's "unconscious" and its "conscious" activities. Eventually scenarios are created and evaluated for each candidate billet and one of them is chosen. Thus we have behavior control via deliberation.

The mediation by the "consciousness" mechanism, as described in the previous paragraph is characteristic of IDA. The principle is that she should use "consciousness" whenever a human detailer would be conscious in the same situation. For example, IDA could readily recover all the needed items from a sailor's personnel record unconsciously with a single behavior stream. But, a human detailer would be conscious of each item individually. Hence, according to our principle, so must IDA be "conscious" of each retrieved personnel data item.

Learning into Associative Memory

Both CMattie and IDA employ sparse distributed memory (SDM) as their major associative memories (Kanerva 1988). SDM is a content addressable memory that, in many ways, is an ideal computational mechanism for use as a long-term associative memory. Being content addressable means that items in memory can be retrieved by using part of their contents as a cue, rather than having to know the item's address in memory.

The inner workings of SDM rely on large binary spaces, that is, spaces of vectors containing only zeros and ones, called bits. These binary vectors, called words, serve as both the addresses and the contents of the memory. The dimension of the space determines the richness of each word. These spaces are typically far too large to implement in an conceivable computer. Approximating the space uniformly with a possible number of actually implemented, hard locations surmounts this difficulty. The number of such hard locations determines the carrying capacity of the memory. Features are represented as one or more bits. Groups of features are concatenated to form a word. When writing a word to memory, a copy of the word is placed in all close enough hard locations. When reading a word, a close enough cue would reach all close enough hard locations and get some sort of aggregate or average out of them. As mentioned above, reading is not always successful. Depending on the cue and the previously written information, among other factors, convergence or divergence during a reading operation may occur. If convergence occurs, the pooled word will be the closest match (with abstraction) of the input reading cue. On the other hand, when divergence occurs, there is no relation -in general- between the input cue and what is retrieved from memory.

SDM is much like human long-term memory. A human often knows what he or she does or doesn't know. If asked for a telephone number I've once known, I may search for it. When asked for one I've never known, an immediate "I don't know" response ensues. SDM makes such decisions based on the speed of initial convergence. The reading of memory in SDM is an iterative process. The cue is used as an address. The content at that address is read as a second address, and so on until convergence, that is, until subsequent contents look alike. If it doesn't quickly converge, an "I don't know" is the response. The "on the tip of my tongue phenomenon" corresponds to the cue having content just at the threshold of convergence. Yet another similarity is the power of rehearsal during which an item would be written many times and, at each of these to a thousand locations That's the "distributed" pare of sparse distributed memory. A well-rehearsed item can be retrieved with smaller cues. Another similarity is forgetting, which would tend to increase over time as a result of other similar writes to memory.

How do the agents use this associative memory? As one example, let's suppose an email message for CMattie arrives, is transferred into the perceptual workspace (working memory), and is descended upon by perceptual codelets looking for words or phrases they recognize. When such are found, nodes in the slipnet (a semantic net type mechanism with activation passing that acts as a perceptual and conceptual knowledge structure) are activated, a message type is selected, and the appropriate template filled. The information thus created from the incoming message is then written to the perception registers in the focus, making it available to the rest of the system.

The contents of the focus are then used as an address to query associative memory. The results of this query, that is, whatever CMattie associates with this incoming information, are written into their own registers in the focus. This may include some emotion and some previous action. Attention codelets then attempt to take this information to "consciousness," along with any discrepancies they may find, such as missing information, conflicting seminar times, etc. Information about the current emotion and the currently executing behavior are written to the focus by appropriate codelets. The current percept, consisting of the incoming information as modified by associations and the current emotion and behavior, are then written to associative memory. Those percepts carry strong emotions are written repeatedly yielding stronger associations. IDA handles perception in much the same way.

Thus one type of associative learning takes place,

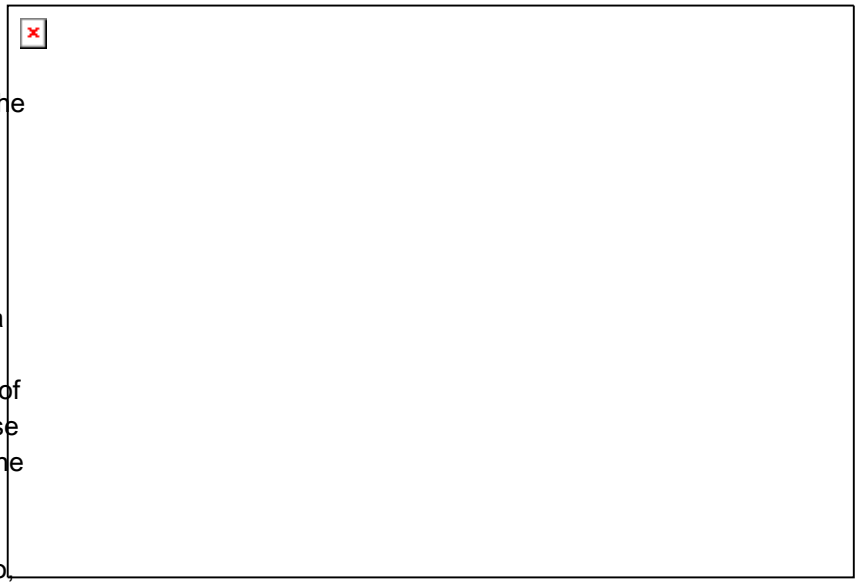
learning into associative memory. For a quite unusual percept, this learning may be only memorization. For more common percepts some sort of generalization takes place as a result of the mechanism of SDM. For example, CMattie learns the usual times of a particular seminar, and will send an inquiry to the organizer should a different time be specified. Due to her much more complex domain, we expect IDA to learn into associative memory in much more sophisticated ways. For example, she may well associate a certain class of sailors, identified by pay grade, job skills, length of obligatory service remaining, etc., to prefer certain types of billets. Thus we've seen one way in which these agents can learn.

Temporal proximity learning

Learning into associative memory as described in the preceding section can be considered a type of declarative learning. We'll now turn to a more procedural learning. Codelets in CMattie (and IDA) participate in a pandemonium theory style organization (Jackson 1987). We'll use Jackson's metaphor. Picture a collection codelets occupying a sports stadium. In the stands are the inactive codelets. This must not be taken too literally. Each of these codelets is alert to conditions that would cause it to become active and join the playing field or, in the case of generator codelets, to instantiate a copy of itself, with variables bound, into the playing field. In the sidelines we find such instantiated codelets who, having responded to a conscious broadcast, now await the activation of their behavior, that is, for their goal context to become dominant. On the playing field we find the active codelets, the codelets that are actively carrying out their functions. Some of these are joined in coalitions. One such coalition should lie in the spotlight of "consciousness". One can think of the playing field as the agent's working memory or, better yet, as the union of her several working memories.

The agent's coalition manager is responsible for forming and tracking coalitions of codelets on the playing field. Such coalitions are initiated on the basis of the mutual *associations* between the member codelets. Since association can both increase and diminish the forming and tracking of coalitions is a dynamic process.

While the existence of a coalition depends on the strengths of the associations between its members, its chance of becoming "conscious" depends on their average *activation*. The agent's spotlight controller is responsible for selecting the coalition with the highest such average to shine upon. Since activations change even more rapidly than associations, the spotlight of "consciousness" can be expected to frequently shift from one coalition to another. The activation of a codelet can be influenced by a higher level concept (slipnet node, behavior), by the current emotion and, in the case of an attention codelet, by its own action. A codelet's activation goes to zero when its task is finished.



Those codelets who share time in the spotlight of "consciousness" have associations between them formed or strengthened, or perhaps diminished if things are going badly. Those codelets sharing time in the playing field also change associations, but at a much lesser rate. Coalitions of highly associated codelets may merge to form higher-level concept codelets. The concept codelet, when active, performs the same actions as do its member codelets combined. This is comparable to chunking in SOAR (Laird et al. 1987).

Thus we've described two different types of temporal proximity, or Hebbian, style learning. As codelets become more or less associated likelihood of their coming to "consciousness" changes, and with

it the likelihood of their together helping to initiate some behavior. These changes in likelihood constitute a kind of procedural learning. When the same coalition of codelets, acting in parallel or in sequence, often produce a useful result, this coalition can be expected to merge into a concept codelet. This merging constitutes a second form of temporal proximity learning.

Learning of emotional influence

In both CMattie and IDA we include mechanisms for emotions (McCauley & Franklin 1998) . CMattie, for example may "experience" such emotions as guilt at not getting an announcement out on time, frustration at not understanding a message, and anxiety at not knowing the speaker and title of an impending seminar. Action selection will be influenced by emotions via their effect on drives, modeling recent work on human action selection (Damasio 1994).

CMattie can "experience" four basic emotions, anger, fear, happiness and sadness. These emotions can vary in intensity as indicated by their activation levels. For example, anger can vary from mild annoyance to rage as its activation rises. A four vector containing the current activations of these four basic emotions represents CMattie's current emotional state. Like humans, there's always some emotional state however slight. Also like humans, her current emotional state is often some complex combination of basic emotions or results from some particular changes in them. The effect of emotions on codelets, drives, etc. varies with their intensity. Fear brought on by an imminent shutdown message might be expected to strengthen CMattie's self-preservation drive resulting in additional activation going from it into the behavior net.

CMattie's emotional codelets serve to change her emotional state. When its preconditions are satisfied, an emotional codelet will enhance or diminish one of the four basic emotions. An emotion can build till saturation occurs. Repeated emotional stimuli result in habituation. Emotion codelets can also combine to implement more complex secondary emotions that act by affecting more than one basic emotion at once. Emotion codelets also serve to enhance or diminish the activation of other codelets. They also act to increase or decrease the strength of drives, thereby influencing CMattie's choice of behaviors.

IDA's emotional system will need to be a good bit more robust than CMattie's. In addition, IDA's emotions will be more tightly integrated with her "consciousness" mechanisms. The first step in this direction will be to meld portions of the emotion and "consciousness" mechanisms by adding to some of the attention codelets the ability to change the emotion vector, and by linking their activation to the amount of emotional change produced by that codelet. The next step provides massive interconnectedness between the emotional mechanisms and the other cognitive areas of the system. A network is built up by connecting the attention/emotion codelets to key behaviors, goals, drives, perception codelets, etc. The links of this network are to have weights and carry activation. Weights will decay with disuse. Each use tends to decrease the decay rate. Weights will increase according to a sigmoidal function of any activation carried over the link, allowing for Hebbian style learning. The product of weight and carried activation is added to the activation already present at the head of the link. Spreading activation then becomes the common currency that integrates the separate modules that use these constructs.

Thus one of the agents, IDA, will be able to learn use emotions to affect various cognitive functions as we humans do.

Metacognitive learning

Metacognition should include knowledge of one's own cognitive processes, and the ability to actively monitor and consciously regulate them. This would require self-monitoring, self-evaluation, and self-regulation. Following Minsky, we'll think of CMattie's "brain" as consisting of two parts, the A-brain and the B-brain (Minsky 1985). The A-brain consists of all the other modules of the agent's architecture. It performs all of her cognitive activities except metacognition. Its environment is the outside world, a dynamic, but limited, real world environment. The B-brain, sitting on top of the A-brain, monitors and regulates it. The B-brain's environment is the A-brain, or more

specifically, the A-brain's activities.

One can look at a metacognitive module as an autonomous agent in its own right. It senses the A-brain's activity and acts upon it over time in pursuit of its own agenda. It's also structurally coupled to its quite restricted environment. Its agenda derives from built in metacognitive drives. One such drive is to interrupt oscillatory behavior. Another such might be to keep the agent more on task, that is, to make it more likely that a behavior stream would carry out to completion. Yet another would push toward efficient allocation of resources.

Unlike the situation in her A-Brain where drives are explicitly represented as part of the behavior net, the agent's metacognitive drives are embodied in fuzzy classifiers (Zhang et al. 1998). The preconditions of such classifiers typically include some specification of an emotional state. Another type of precondition may involve the number of email messages in the incoming queue, or the number of instantiated behavior streams, or the amount of memory space they are using.

How does the metacognition module influence the agent's behavior? In all cases it relies on its inner percepts of the A-brain. Oscillatory behavior might occur as the perceptual mechanism goes back and forth between two message types unable to decide on either. Metacognition might then send additional activation to one message type node in the slipnet, effectively forcing a decision, even a wrong one. The metacognition module can also affect the agent's behavior by tuning global parameters, for example in the behavior net. This kind of tuning could serve to keep her more on task, by increasing the parameter that controls the amount of activation a drive pumps into its behavior streams. Or, it could make her more thoughtful by increasing the threshold for executing behaviors. Finally, metacognition may be concerned with high-level allocation of resources. For example, memory might be shifted from, say, a workspace (part of working memory) to the behavior net to accommodate a shortage of space there.

Metacognition in CMattie is implemented as a classifier system in order that it may learn. Learning actions always requires feedback on the results of prior actions. The Evaluator submodule is implemented by a reinforcement learning algorithm (Barto et al. 1981) that assigns reward or punishment to classifiers based on the next inner percept. It also uses a reinforcement distribution algorithm (Bonarini 1997) to distribute credit among the classifiers. The more common bucket brigade algorithm (Holland et al. 1978) is not used since sequences of actions are not typically required of metacognition in these agents. When things are not going too well over a period of time, learning occurs via a genetic algorithm (Holland 1975) acting to produce new classifiers. Thus we have yet another form of learning in "conscious" software agents.

Learning by being told

Our "conscious" software agents will learn concepts into their perceptual mechanisms, that is, they'll learns new slipnet nodes and links, and new perceptual codelets (Ramamurthy et al. 1998, Bogner et al. in press). This learning takes place by modifying what's known, existing nodes, links and codelets, using case based reasoning (Kolodner 1993). In CMattie the impetus for such learning comes from messages from a seminar organizer informing her that she has mishandled a previous message. An interchange of email messages between CMattie and the organizer may eventually lead to her learning a new concept. We'll trace a hypothetical scenario for such learning.

Suppose CMattie receives an announcement of a dissertation defense to be held at a certain place and time with a certain speaker and title. She would most likely treat this as a speaker-topic message for a new seminar. This misunderstanding is disseminated through "consciousness", leading to an acknowledgement to the sender stating that she is initializing a new seminar called ``Dissertation Defense Seminar'' with the sender as its organizer. This acknowledgement may well elicit a negative response from the sender. CMattie has slipnet nodes, including a message type, codelets and behaviors to help deal with such a situation. Such a negative response may start a "conversation" between CMattie and the sender. During this interchange, CMattie learns that a dissertation defense is similar to a seminar, but with slightly different features. In this case, the periodicity feature has a different value. The email conversation, stripped of headers and pleasantries might go something like this:

Sender: *It's not a dissertation defense seminar, just a dissertation defense.*
CMattie: *What's a dissertation defense?*
Sender: *It's like a seminar but only happens*

irregularly.

CMattie can trace the thread of the conversation via her episodic memory (case-based memory in CMattie, perhaps SDM in IDA). CMattie has codelets that recognize words associated with features. Thus she should recognize "irregularly" as having a certain meaning with regard to periodicity. At this point, case based reasoning comes into play, allowing the creation of a new slipnet node for dissertation defense with features the same as those of the seminar node except that periodicity is now fixed at "irregular." Links are also put in place with destinations similar to those of the seminar node. A new message type node is also created, along with its links. Finally the needed new codelets are created, modeled after the old codelets associated with the seminar node. Case based reasoning has solved the problem by first identifying the solution to the most similar prior problem, and then modifying it to solve the new one. (In order for this to work, initial cases have to be included in case based memory at startup.) A new concept has been learned to the extent that CMattie needs to learn it. She learned it by being told.

Behavioral learning occurs quite similarly. CMattie's behavioral learning mechanism, again case based, takes note of the changes wrought in the slipnet and deduces needed changes in behavior. This leads to new behavior streams and new codelets to support them. If CMattie initially gets things wrong, another interchange with the appropriate seminar organizer may ensue. Eventually, CMattie will learn an acceptable behavior for a dissertation defense. Note that we've described what is essentially a one-shot learning. Though we might consider this reinforcement learning, it would be a stretch. This global learning is quite different from the local learning common in new AI systems, such as in neural net or reinforcement learning. Again, it is learning by being told.

Learning by imitation

Thought still in the speculation stage, it seems possible that IDA will have to learn about a particular community of sailors, say cooks, machinist mates, or sonar technicians, by watching a human detail in action dealing with them. Though there has been some discussion of adding this type of learning to the system, as yet no decision has been taken. In fact, no mechanism for this kind of learning has been seriously proposed. At this writing we are on the verge of initiating a literature search. Learning by imitation is quite common in humans, and may well be needed in "conscious" software agents. This may be a part on a necessary development on the part of more sophisticated "conscious" software agents.

Implications for human learning

In the previous sections we've seen descriptions of several quite distinct types of learning available to "conscious" software agents. Declarative learning occurs in SDM and episodic memory implemented as case-based memory. Percepts from the focus being stored in the main associative memory constitute declarative learning. Not only is the content of the percept learned, but relationships with other items of memory and some generalization also. Each new case stored constitutes declarative learning. Declarative learning also results from the learning of new concepts by being told by seminar organizers. We suspect that human declarative learning also occurs in each of these forms, each with a somewhat different mechanism.

Procedural learning in these agents also occurs in several forms. Gradual procedural learning takes place when associations are strengthened between codelets that are "conscious," or even active, together. The chunking of codelets is another form of procedural learning, as is the learning of new behaviors through email interactions with organizers. We suspect that humans are capable of each of these modes of procedural learning and that they each require a different mechanism. Of course, much important human learning, of language for example, has both a declarative and a procedural component.

A basic tenet of global workspace theory says that consciousness is sufficient for learning. It certainly is true in our agents. The contents of "consciousness" are routinely written to associative memory. Is consciousness also necessary for learning? The learning of new associations between codelets and adjustments to such associations

happens when their contents become "conscious." But it also occurs to a lesser extent when the codelets are active together, but unconscious. Our chunking mechanism also does not routinely come to consciousness, though a newly learned chunk may well have been "conscious." This seems to suggest that some procedural learning, some gradual improvement of skills may occur unconsciously with practice.

It also seems that our metacognitive learning occurs unconsciously. This must be at least partly a flaw in our model, since at least some metacognitive learning in humans happens consciously. We also suspect that some learning of weights in emotion networks of these agents will occur unconsciously. This seems quite possibly true of humans as well.

References

Baars, B. J. 1988. *A Cognitive Theory of Consciousness*. Cambridge: Cambridge University Press.

Baars, B. J. 1997. *In the Theater of Consciousness*. Oxford: Oxford University Press.

Barsalou, L. W. 1999. Perceptual symbol systems. *Behavioral and Brain Sciences* 22:577–609.

Barto, A. G., R. S. Sutton, and P. S. Brouwer. 1981. Associative Search Network: a Reinforcement Learning Associative Memory. *Biological Cybernetics* 40:201–211.

Bogner, M. 1999. Realizing "consciousness" in software agents. Ph.D. Dissertation. University of Memphis.

Bogner, M., U. Ramamurthy, and S. Franklin. in press. Consciousness" and Conceptual Learning in a Socially Situated Agent. In *Human Cognition and Social Agent Technology*, ed. K. Dautenhahn. Amsterdam: John Benjamins.

Bonarini, A. 1997. *Anytime Learning and Adaptation of Structured Fuzzy Behaviors. Adaptive Behavior Volume 5*. Cambridge MA: The MIT Press.

Damasio, A. R. 1994. *Descartes' Error*. New York: Gosset; Putnam Press.

Edelman, G. M. 1987. *Neural Darwinism*. New York: Basic Books.

Franklin, S. 1995. *Artificial Minds*. Cambridge MA: MIT Press.

Franklin, S. 1997. Autonomous Agents as Embodied AI. *Cybernetics and Systems* 28:499–520.

Franklin, S., and A. C. Graesser. 1997. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Intelligent Agents III*. Berlin: Springer Verlag.

Franklin, S., A. Kelemen, and L. McCauley; 1998; IDA: A Cognitive Agent Architecture. IEEE Conf on Systems, Man and Cybernetics.

Hofstadter, R. D., and Mitchell M... 1994. *The Copycat Project: A model of mental fluidity and analogy-making. In: Advances in connectionist and neural computation theory, Vol. 2: Analogical connections, eds. K. J. Holyoak & J. A. Barnden*. Norwood N.J.: Ablex.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Holland, J. H. 1986. A Mathematical Framework for Studying Learning in Classifier Systems. In *Evolution, Games and Learning: Models for Adaption in Machine and Nature*, vol. al, Amsterdam, ed. D. Farmer. : North-Holland.

Holland, J. H., H. J., and Reitman J. S... 1978. *Cognitive Systems Based on Adaptive Algorithms. In: Pattern Directed Inference Systems (pp. 313 -329), eds. D. A. Waterman & F. Hayey-Roth*. New York: Academic Press.

Jackson, J. V. 1987. Idea for a Mind. *Siggart* Newsletter, 181:23–26.

Kanerva, P. 1988. *Sparse Distributed Memory*. Cambridge MA: The MIT Press.

Kolodner, J. 1993. *Case-Based Reasoning*. : Morgan Kaufman.

Laird, E. J., Newell A., and Rosenbloom P. S... 1987. SOAR: An Architecture for General Intelligence. *Artificial Intelligence* 33:1–64.

Maes, P. 1990. How to do the right thing. *Connection Science* 1:3.

Maturana, R. H., and F. J. Varela. 1980. *Autopoiesis and Cognition: The Realization of the Living, Dordrecht*. Netherlands: Reidel.

Maturana, H. R. 1975. The Organization of the Living: A Theory of the Living Organization. *International Journal of Man-Machine Studies* 7:313–332.

McCauley, T. L., and S. Franklin; 1998. An Architecture for Emotion. AAAI Fall Symposium Emotional and Intelligent: The Tangled Knot of

Cognition"; AAAI; Orlando, FL.
Minsky, M. 1985. *The Society of Mind*. New York: Simon and Schuster.
Ornstein, R. 1986. *Multimind*. Boston: Houghton Mifflin.
Ramamurthy, U., S. Franklin, and A. Negatu. 1998. Learning Concepts in Software Agents. In *From animals to animats 5: Proceedings of The Fifth International Conference on Simulation of Adaptive Behavior*, ed. R. Pfeifer, B. Blumberg, J.-A. Meyer , and S. W. Wilson. Cambridge,Mass: MIT Press.
Sloman, A. 1987. Motives Mechanisms Emotions. *Cognition and Emotion* 1:217–234.
Zhang, Z., D. Dasgupta, and S. Franklin. 1998. Metacognition in Software Agents using Classifier Systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, Wisconsin .