Franklin, S. 2000. Deliberation and Voluntary Action in 'Conscious' Software Agents. *Neural Network World* 10:505–521.

Deliberation and Voluntary Action 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 briefly the architecture and mechanisms of a "conscious" software agent named IDA who is to act as a type of employment agent. We go on to show how this architecture and these mechanisms allow IDA to deliberate much as we humans do and to make voluntary choices of actions. The technology described, if successful, should allow the automation of the work of many human information agents. The IDA model also makes, hopefully testable, hypotheses about how human deliberation and volition.

Deliberation

In outlining an architecture for cognition Sloman postulates three types of processes, the reactive processes, the deliberative processes, and the meta-management processes (1999) (see Figure 1). It's the deliberative processes that will concern us here.

We humans, when faced with a goal or problem often imagine possible plans for achieving the goal or for solving the problem. These trial plans or solutions typically take the form of scenarios of various kinds, say the steps to a solution, the route to a destination, or a single image scenario as a possible solution to a design problem. The essence of deliberation consists of the creation such scenarios and the choosing among them, discarding some, acting on others. This ability to mentally try out plans or solutions without acting upon them allows us to often avoid actions that may be ineffective or even dangerous. This internal virtual reality has been, perhaps, the major factor in our success as a species.

Evaluation of scenarios plays a critical role in deliberation. This is accomplished with the aid of both reason and emotion. Reason allows us to more or less accurately trace cause and effect in the service of predicting the consequences of possible actions. The emotions provide the values that allow us to decide whether or not we like those consequences. This process enable us to choose among the various scenarios.

Humans can deliberate. No doubt members of some few other species can also. But, what about machines? This paper is devoted to a description of a software agent that can deliberate, though not nearly as well as we humans.

Autonomous Agents

Artificial intelligence pursues the twin goals of understanding human intelligence and of producing intelligent software and/or artifacts. Designing, implementing and experimenting

[1] Supported in part 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, Arpad Kelemen, Ravikumar Kondadadi, Irina Makkaveeva, Lee McCauley, Aregahegn Negatu, Hongjun Song, Allexei Stoliartchouk, Uma Ramamurthy, and Zhaohua Zhang

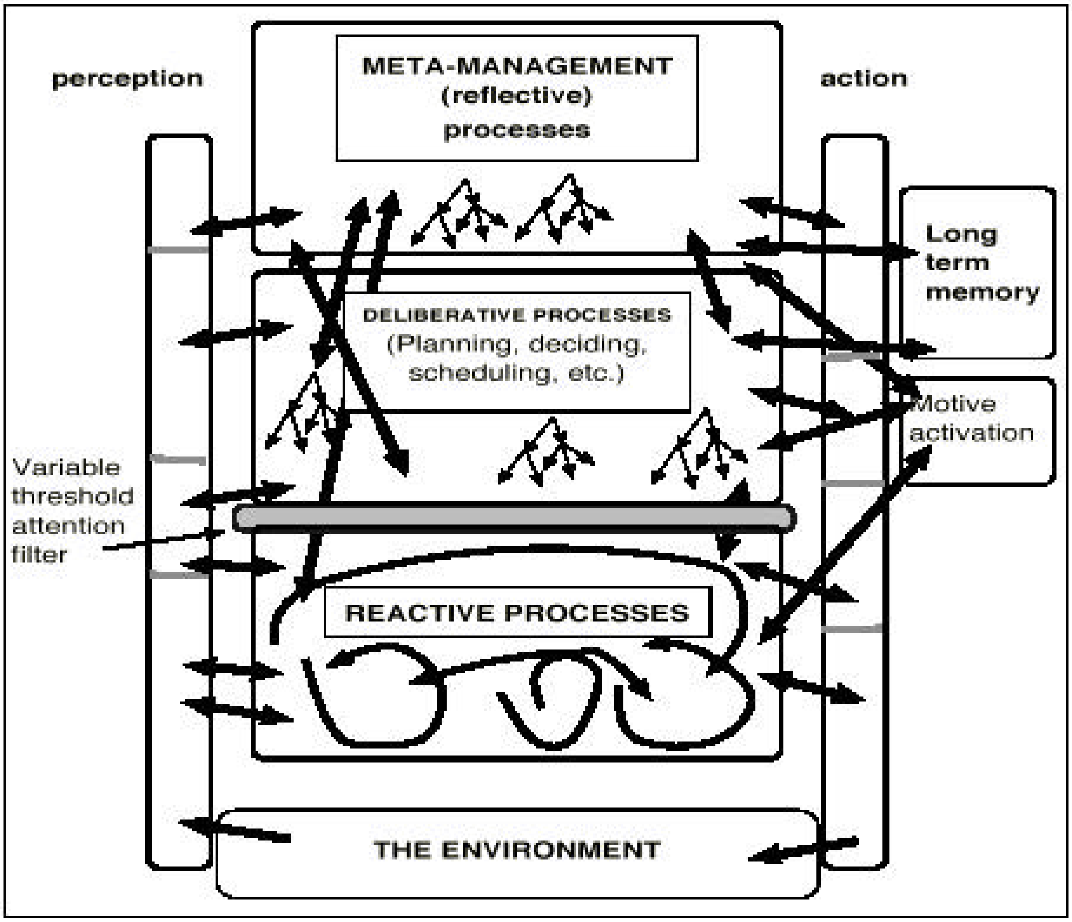

Figure 1. Sloman's Cognitive Architecture

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 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

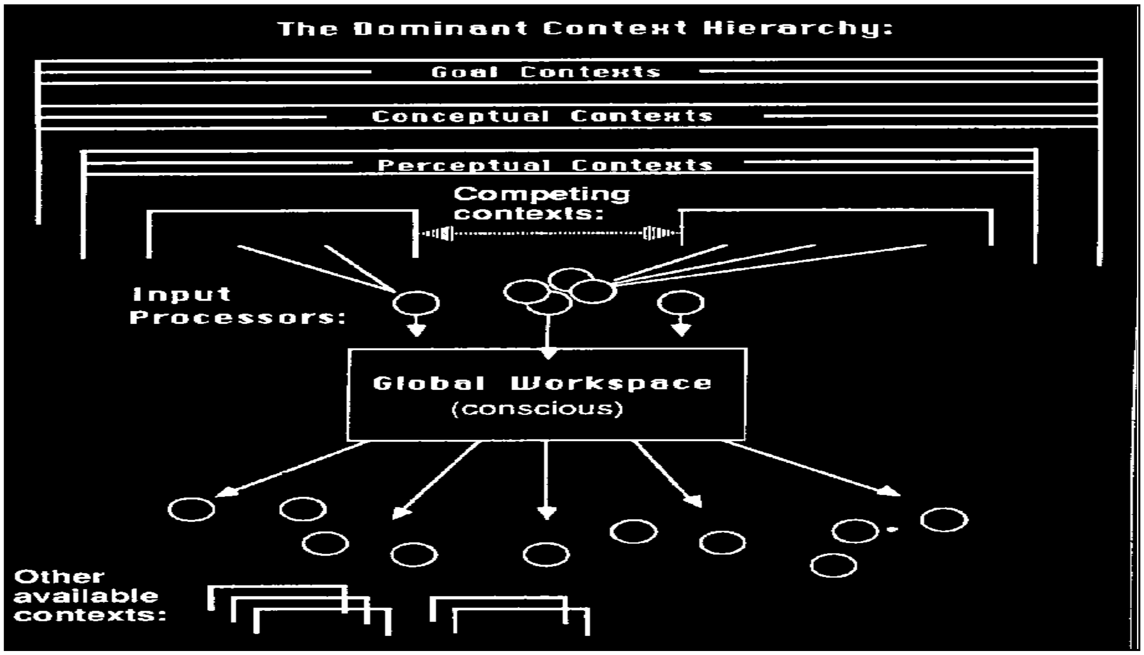

Figure 2. Global Workspace Theory

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 novel 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.

This theory offers an explanation for consciousness being serial in nature rather than parallel as is common in the rest of the nervous system. Messages broadcast in parallel would tend to overwrite one another making understanding difficult. It similarly explains the limited capacity of consciousness as opposed to the huge capacity typical of long-term memory and other parts of the nervous system. Large messages would be overwhelming to small, special-purpose processors.

All this activity of processors takes place under the auspices of contexts (see Figure 2): 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 or phenomenal consciousness 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).

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 almost 300 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

The IDA architecture consists of both an abstract level (containing such entities as behaviors, message type nodes, metacognitive actions, etc.), and a lower, more specific level (implemented by small pieces of code). At the higher level the architecture is quite modular with module names often borrowed from psychology (see Figure 3). There are modules for Perception, Action Selection, Associative , Episodic Memory, Emotions, Metacognition, Learning, Constraint Satisfaction, Language Generation, Deliberation, and “Consciousness.” Many of their mechanisms were inspired by ideas from the “new AI” (the copycat architecture (Mitchell 1993, Hofstadter & Mitchell 1994, Sloman 1999), behavior nets (Maes 1989), sparse distributed memory (Kanerva 1988),pandemonium theory (Jackson 1987), and fuzzy classifier systems (Zadeh 1965, Holland 1986)). Others come from more classical AI (case-based reasoning (Kolodner 1993)).

In the lower level of the IDA architecture 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. Most of these codelets subserve some high level entity such as a behavior or a slipnet node or a metacognitive action. Some codelets work on their own performing such tasks as watching for incoming email and instantiating goal structures. An important type of the latter is the attention codelets who serve to bring information to “consciousness.” Codelets do almost all the work, making IDA is a multi-agent system.

Perception

IDA senses text, not imbued with meaning, but as primitive sensation as for example the pattern of activity on the rods and cones of the retina. This text may come from email messages, a chat room environment, or from a database record. Her perception module (much like that of an

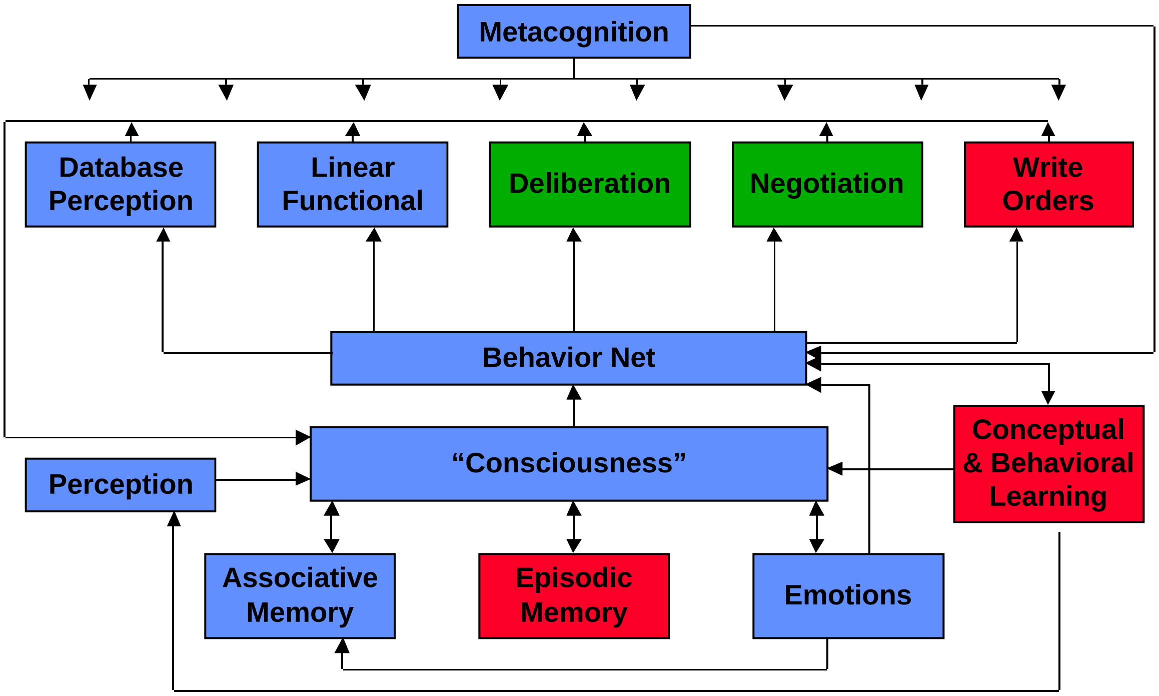

Figure 3. The IDA Architecture

earlier such agent (Zhang et al. 1998b)), employs analysis of surface features for natural language understanding (Allen 1995). It partially implements perceptual symbol system theory (Barsalou 1999), which is used as a guide. Its underlying mechanism constitutes a portion of the Copycat architecture (Hofstadter & Mitchell 1994). The perceptual/conceptual knowledge base of IDA takes the form of a semantic net with activation passing called the slipnet (see Figure 4). The name is taken from the Copycat architecture. 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. The slipnet embodies the perceptual contexts and some conceptual contexts from global workspace theory. There's a horde of perceptual codelets that descend on an incoming message, looking for words or phrases they recognize. When such are found, appropriate nodes in the slipnet are activated, This activation passes around the net until it settles. An idea type node (or several) is selected by its high activation, and the appropriate template(s) filled by codelets with selected items from the message. The information thus created from the incoming message is then written to the perception registers in the workspace (to be described below), making it available to the rest of the system.

The results of this process, information created by the agent for its own use (Franklin 1995), are written to the workspace (short term memory not to be confused with Baars' global workspace). Almost all IDA's modules either write to the workspace, read from it, or both. The focus, to be describe below, is part of this workspace.

Associative Memory

IDA employs sparse distributed memory (SDM) as her major associative memory (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.

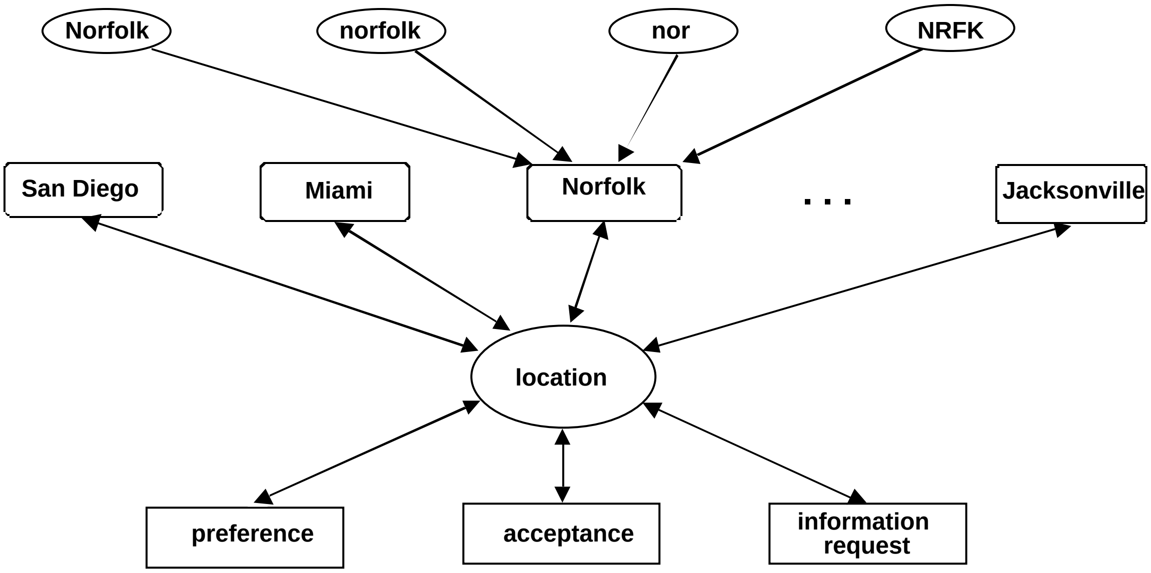

Figure 4. A Small Portion of IDA's Slipnet

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 does IDA use this associative memory? Reads and writes to and from associative memory are accomplished through a gateway with the workspace called the focus (see Figure 5). When any item is written to the workspace, another copy is written to the read registers of the focus. The contents of these read registers of the focus are then used as an address to query associative memory. The results of this query, that is, whatever IDA associates with this incoming information, are written into their own registers in the focus. This may include some emotion and some action previously taken. Thus associations with any incoming information, either from the

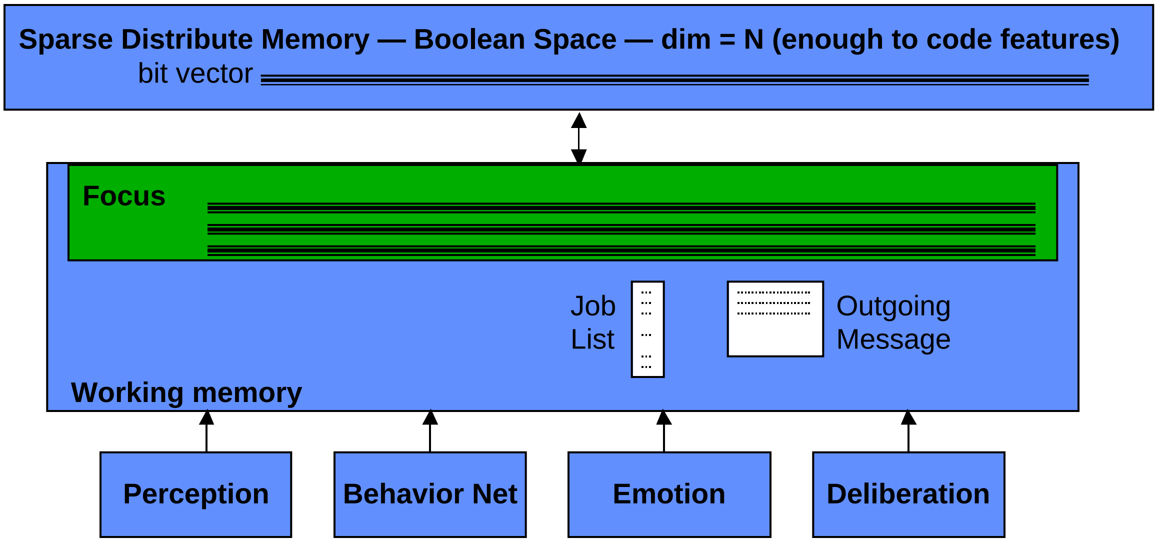

Figure 5. IDA's Associative Memory and Workspace

outside world, or from some part of IDA herself, are immediately available. Writes to associative memory are made latter and will be described below.

"Consciousness"

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. 2000). Attention codelets have the task of bringing information to "consciousness." 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 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.

Action Selection

IDA depends on expansion of the idea of a behavior net (Maes 1989) for high-level action selection in the service of built-in drives (see Figure 6). She 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 intended to measure the behavior's relevance to both the current environment (external and internal) and its ability to help satisfy the various drives it serves. Each behavior occupies a node in a digraph

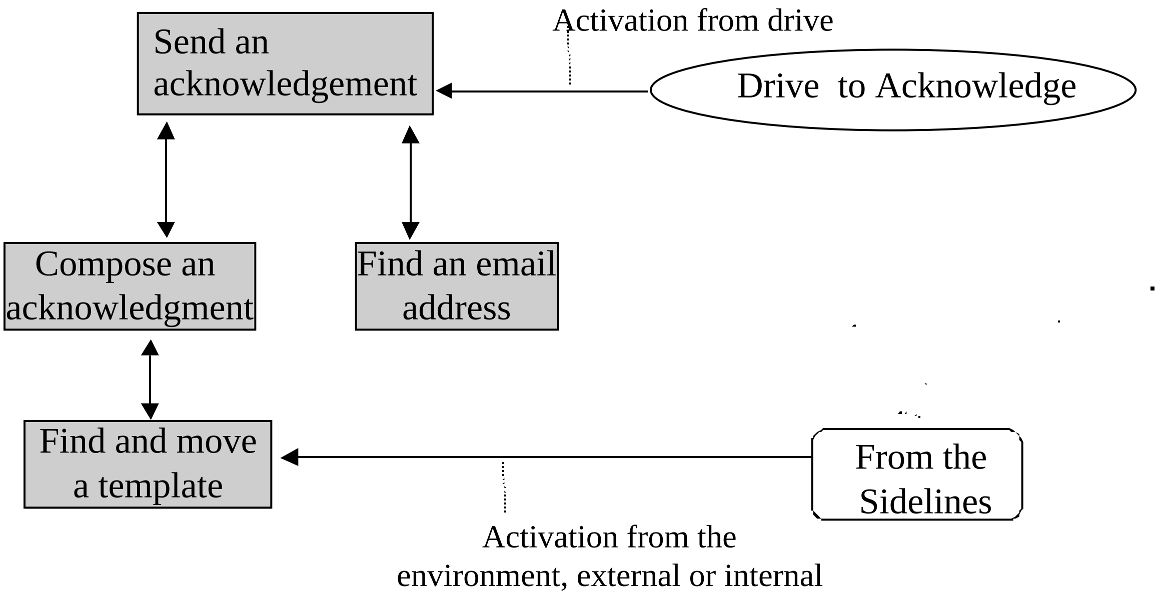

Figure 6. A Simple Goal Structure

(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 external 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 help to 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.

IDA’s behavior net acts in consort with her “consciousness” mechanism to select actions. Here’s how it works. Suppose some piece of information is written to the workspace by perception or some other module. Vigilant attention codelet watch both it and the resulting associations. One of these attention codelets may decide that this information should be acted upon. This codelet would then attempt to take the information to “consciousness,” perhaps along with any discrepancies it may find with the help of associations. The attention codelet and the needed information carrying codelets become active. If the attempt is successful, the coalition manager makes a coalition of them, the spotlight controller eventually selects that coalition, and the contents of the coalition are broadcast to all the codelets. In response to the broadcast, appropriate behavior priming codelets perform three tasks: 1) if it’s not already there, an appropriate goal structure is instantiated in the behavior net. 2) wherever possible the codelets bind variables in the behaviors of that structure. 3) the codelets send activation to the currently appropriate behavior of the structure. Eventually that behavior is chosen to be acted upon. At this point, information about the current emotion and the currently executing behavior are written to the focus by the behavior codelets associated with the chosen behavior. The current contents of the write registers in the focus are then written to associative memory. The rest of the behavior codelet associated with the chosen behavior then perform their tasks. An action has been selected and carried out by means of collaboration between “consciousness” and the behavior net.

Constraint satisfaction

At the heart of IDA’s task of finding new jobs for sailors lies the issue of constraint satisfaction. Not only must IDA look out for the needs of the sailor, she must also see that the requirements for individual jobs are met, and simultaneously adhere to the policies of the Navy. Sailors tend to stay in the Navy when they are satisfied with their job assignment, and to leave at the end of an enlistment when they aren’t. Thus, keeping the sailors happy is an issue of central concern to the Navy. Each individual job presents its own constraints in terms of job qualifications, location, sea or shore duty, time of arrival, etc. Finally, the policies of the Navy must be adhered to. For example, a sailor finishing shore duty should be next assigned to sea duty. Taking such issues into consideration, IDA’s constraint satisfaction module is designed to provide a numerical measure of the fitness of for a particular job for a particular sailor. Here’s how it is to work.

Given a specified issue such as sailor preference, a particular Navy policy or specific job requirement, referred to as j for short, we define a function x_j that provides a numerical measure of the fitness of this job for this sailor with respect to this particular issue. For example, suppose the issue j is the one that says a sailor may be assigned to a job requiring a certain paygrade, if his or her paygrade is no more than one more or less. Here we might define x_j as follows: $x_j = 1$ if the sailor has the specified paygrade, $x_j = 0.5$ if the sailor’s paygrade is one more or less than that specified, and $x_j = 0$ otherwise. This would provide the desired numerical measure of fitness with respect to this particular policy.

Having chosen in consultation with human detailers a collection of issues to be considered by IDA, we must create such a fitness function x_j for each of them. Computationally, the functions must be quite diverse. Most would take their input from information from the sailor’s personnel record or from the job requisition list that has already been written to the workspace. As in the example above, the numerical values of the functions must lie between 0 and 1.

With these functions in hand, IDA can tell how suitable a particular job was for a specified sailor with respect to any given issue. But, what about with respect to all of them? How can we measure the overall suitability of this job for this sailor? How can we combine the individual fitness measures associated with the individual issues? What’s the common currency?

What we need is, for each issue j, a numerical measure a_j of the relative importance of that issue with respect to all the other issues. Such measures can be determined in consultation with expert human detailers using statistical methods. They may also be approximated from data concerning actual assignments of sailors to jobs by human detailers. Some combination of these two methods may contribute to a more accurate choice of the a_j. Each a_j should also lie between 0 and 1, and their sum should be 1. Each a_j will be used to weight the result of the corresponding function x_j that measures the suitability of the given job for the sailor in question with respect to the issue j. Thus the weighted sum of the x_j, $\Sigma a_j x_j$, will give the required total fitness measure with respect to all the issues. This is our common currency. IDA now has a measure of the fitness of a particular billet for the sailor in question.

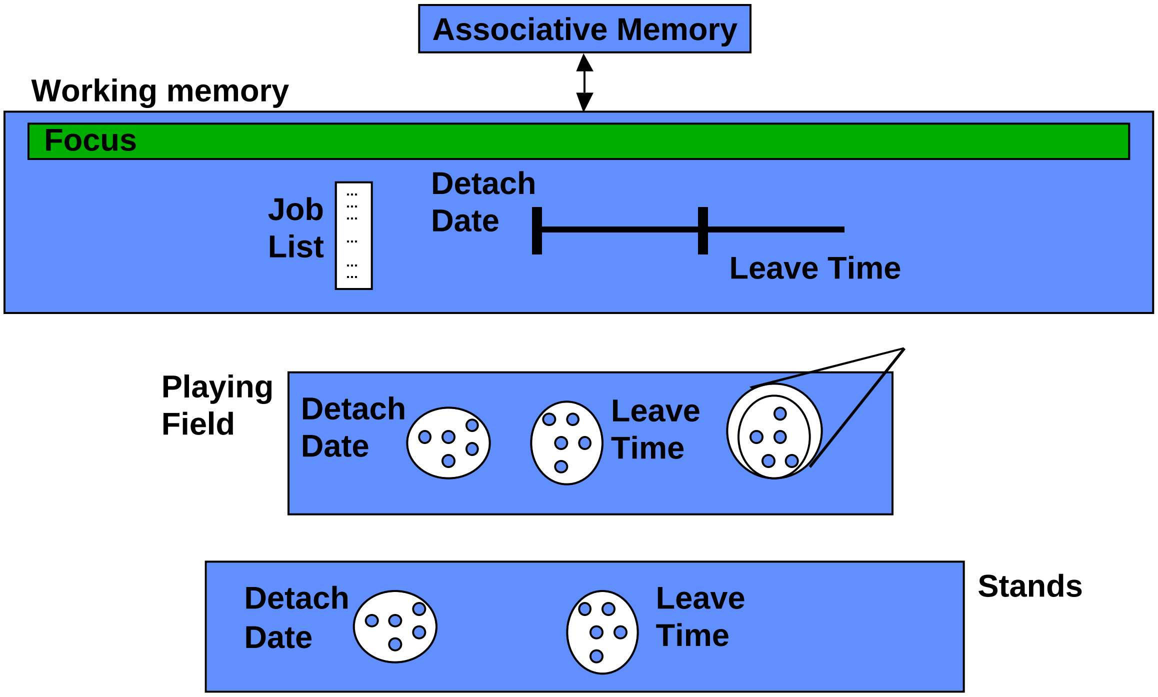

Figure 7. IDA's Deliberation in Action

Deliberation in Action

IDA's relatively complex domain 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 order to determine whether or not a given job can meet joint temporal constraints such as the sailor's projected rotation date (PRD) and the take-up month (TUM) of the billet. And, a sailor to be assigned to a certain ship had best arrive before the ship sails. If this can't be accomplished, some other assignment must be made. 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. There's travel time to be figured in. Such scenarios are valued on how well they fit these temporal constraints as well as on moving and training costs.

Scenarios are composed of scenes. IDA's scenes are organized around events. Each scene may, in theory, require objects, actors, concepts, relations, and schema represented by frames. In practice in this domain they are not all that complicated involving mostly dates and time intervals. Scenarios are constructed in the computational workspace described above, which corresponds to working memory in humans. The scenes are strung together to form scenarios. The work is done by deliberation codelets. Evaluation of scenarios is also done by codelets.

Here we'll describe the beginnings of the construction of such a scenario. The process described is common to almost all of IDA's action selection. Four type of codelets are involved, all of whom we've met before. The attention codelets serve to actuate the bringing of information from the workspace to consciousness. Information codelets actually carry most of the information during this process. Behavior priming codelets respond to "conscious" broadcasts instantiating goal structures in the behavior net (if not already there), binding variables within individual behaviors, and sending activation to relevant behaviors. Behavior codelets execute task needed to implement the behavior they serve when that behavior is selected by the behavior net to be acted upon.

On to scenario building. At this point in IDA's search for a job for the given sailor, a list of jobs coarsely selected from the current requisition list are already in the workspace. One by one they've been acted upon by the constraint satisfaction module resulting in an attached numerical fitness value. Some attention codelet notices that the last fitness value has been written next to its job. This is its cue to begin the scenario building process.

This attention codelet selects a job for the scenario (typically the one with the highest fitness) and recruits information codelets to carry specific information about the job. All these codelets are now active and, thus, available to the coalition manager. Typically they will comprise a coalition. If (or when) this coalition has sufficient activation, the spotlight will shine upon it. It's contents are then broadcast to all the other codelets. "Consciousness" has done its work.

Appropriate behavior priming codelets respond. Some extract information from the broadcast. Others know which goal structure to instantiate, in this case a create-scenario goal structure. The goal structure is instantiated, the behavior variables are bound where possible, and activation is sent to the behavior that should begin the scenario creation. Eventually, that behavior will be selected by the behavior net to be acted upon. It's codelets will then execute their tasks, writing the first scene of the scenario to the workspace. In this case the first scene with consist of a month during which the sailor is to detach from his current job.

Now the process starts again. A different attention codelet notices that the detach month of a scenario has been written in the workspace. It's its turn to recruit information to help build the next scene. The action of the "consciousness" module, the behavior priming codelets, the behavior net and the behavior codelets proceed as before. (Note that the behavior priming codelets will not have to instantiate a new goal structure this time.) All this results in a leave time period (typically thirty days) being written to the workspace as the second scene in the scenario.

The same process continues over and over again writing a scene for travel time, for proceed time (if needed and which I won't explain) for the beginning date of a training class (if needed), for the time interval of the class (if needed), and for the report-no-later-than date. The next scene written computes the gap, which depends on the relationship between the report date and the take up month. If the former is within the later, the gap is zero, otherwise more. The computation is performed by a behavior codelet.

At this point several attention codelets may vie for the next broadcast. The create-scenario attention codelet will have chosen another job for the next scenario and recruited information codelets. If the gap is non-zero, the adjust-the-gap attention codelet will try to instigate the building of a new scenario for the same job with a different detach date that may produce a smaller gap. Or, a proposer attention codelet may like this job and want to propose that it be one of those offered to the sailor (more of this in the next section).

Our contention, and that of global workspace theory, is that we humans deliberate via just such a process. From a computer science point of view, this process is grossly inefficient. How can we justify it in a software agent? In this agent we can't, since IDA's deliberation is routine in that the same actions are performed over and over, and are "well understood" by the agent. In a more challenging domain, an agent may face novel or problematic situations in which the best course of action is not at all apparent. In such a case our inefficient process may allow the agent to deliberate about the probable results of novel courses of action, and to choose a promising one without actually acting on the others. How is such a choice made? This brings us to voluntary action.

Voluntary action

We humans most often select actions subconsciously, that is without conscious thought about which action to take. Sometimes when I speak, I'm surprised at what comes out. But we humans also make voluntary choices of action, often as a result of deliberation. Baars argues that voluntary choice is the same as conscious choice (1997, p. 131). We must carefully distinguish between being conscious of the results of an action, and consciously deciding to take that action, that is, being conscious of the decision. I am typically conscious of my speech (the results of actions) but not typically conscious of the decision to speak. However, sometimes, as in a formal meeting, I may consciously decide to speak and then do so. The decision itself becomes conscious. It's the latter case that constitutes voluntary action.

Long back, William James proposed the ideomotor theory of voluntary action (James 1890). James suggests that any idea (internal proposal) for an action that comes to mind (to consciousness) is acted upon unless it provokes some opposing idea or some counter proposal. He speaks at length of the case of deciding to get out of a warm bed into an unheated room in the dead of winter. "This case seems to me to contain in miniature form the data for an entire psychology of volition." Global workspace theory adopts James' ideomotor theory as is, and provides a functional architecture for it (Baars 1997 , Chapter 6). Here we provide an underlying mechanism that implements that theory of volition and its architecture in the software agent IDA.

Though voluntary action is often deliberative, it can also be reactive in the sense of Sloman (1999), who allows for the possibility or the action selection mechanism being quite complex. Suppose that, while sitting on the couch in my living room, I decide I'd like a cup of coffee and thereafter head for the kitchen to get it. The decision may well have been taken voluntarily, that is, consciously, without my having deliberated about it by considering alternatives and choosing among them. Voluntary actions may also be taken metacognitively (by Sloman's meta-management processes). For example, I might consciously decide to be more patient in the future with my young son. That would be a voluntary metacognitive decision. The IDA model includes a metacognition module that's not discussed in this paper (Zhang et al. 1998a).

But, what about action selection decisions in IDA? Are they voluntary or not? Both kinds occur. When IDA reads as sailor's projected rotation date from the personnel database, she formulates and transmits a query to the database and accepts its response. The decision to make the query, as well as its formulation and transmission, is done unconsciously. The results of the query, the date itself, does come to "consciousness." This situation is analogous to that of almost all human actions. On the other hand, IDA performs at least one voluntary action, that of choosing a job or two or occasionally three to offer a sailor. How is this done?

In the situation in which this voluntary action occurs, at least one scenario has been successfully constructed in the workspace as described in the previous section. The players in this decision making process include several proposing attention codelets and a timekeeper codelet. A proposing attention codelet's task is to propose that a certain job be offered to the sailor. This is accomplished by it bringing information about itself and about the proposed job to "consciousness" so that the timekeeper codelet can know of it. This proposing attention codelet (and its brethren) choose a job to propose on the basis of its particular pattern of preferences. The preferences include several different issues with differing weights assigned to each. The issues typically include priority (stated on the job requisition list), gap, cost of the move, fitness value, and others.

For example, our proposing attention codelet may place great weight on low moving cost, some weight on fitness value, and little weight on the others. This codelet may propose the second job on the scenario list because of its low cost and high fitness, in spite of low priority and a sizable gap. What happens then? There are several possibilities. If no other proposing attention codelet objects (by bringing itself to "consciousness" with an objecting message) and no other such codelet proposes a different job within a span of time kept by the timekeeper codelet, the timekeeper codelet will mark the proposed job as being one to be offered. If an objection or a new proposal is made in a timely fashion, it will not do so.

Two proposing attention codelets may well alternatively propose the same two jobs several times. What keeps IDA from oscillating between them forever? There are three possibilities. The second time a codelet proposes the same job it carries less activation and so has less chance of being selected for the spotlight of "consciousness." Also, the timekeeper loses patience as the process continues, thereby diminishing the time span required for a decision. Finally, the metacognitive module watches the whole process and intervenes if things get too bad.

A job proposal may also alternate with an objection, rather than with another proposal, with the same kinds of consequences. These occurrences may also be interspersed with the creation of new scenarios. If a job is proposed but objected to, and no other is proposed, the scenario building may be expected to continue yielding the possibility of finding a job that can be agreed upon.

We hypothesize that this procedure mimics the way humans make such decisions. It provides a mechanism for voluntary action.

Conclusion

The introduction of deliberative and volitional actions on the part of software agents not only contributes to the modeling of human cognition, but also potentially allows the automation of human information agent. Such human agents must typically possess skills for reading databases, communicating in natural language, adhering to policies, negotiating and decision making. These are precisely IDA's skills. The technology being developed for IDA, if successful, should permit the development of agents that perform all the tasks of human information agents such as insurance agents, travel agents, voter registrars, mail-order service clerks, telephone information operators, employment agents, AAA route planners, customer service agents, and bank loan officers. Of course, each of these is a knowledge intensive task and, thus, presents a knowledge engineering problem for the would be developer. Another possibility is to provide a developmental period for such an agent (Franklin 2000).

References

Allen, J. J. 1995. *Natural Language Understanding*. Redwood City CA: Benjamin/Cummings; Benjamin; Cummings.

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.

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

Bogner, M., U. Ramamurthy, and S. Franklin. 2000. Consciousness" and Conceptual Learning in a Socially Situated Agent. In *Human Cognition and Social Agent Technology*, ed. K. Dautenhahn. Amsterdam: John Benjamins.

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. 2000. Learning in "Conscious" Software Agents. In *Workshop on Development and Learning*. Michigan State University; East Lansing, Michigan, USA: NSF; DARPA; April 5-7, 2000.

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. In *IEEE Conf on Systems, Man and Cybernetics*. : IEEE Press.

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

Holland, J. H. 1986. A Mathematical Framework for Studying Learning in Classifier Systems. *Physica* 22 D:307–317. (Also in Evolution, Games and

Learning. Farmer, J. D., Lapedes, A., Packard, N. H., and Wendroff, B. (eds.). NorthHolland (Amsterdam))

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

James, W. 1890. *The Principles of Psychology*. Cambridge, MA: Harvard University Press.

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

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

Maes, P. 1989. How to do the right thing. *Connection Science* 1:291–323.

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.

Minsky, M. 1985. *The Society of Mind*. New York: Simon and Schuster.

Mitchell, M. 1993. *Analogy-Making as Perception*. Cambridge MA: The MIT Press.

Ornstein, R. 1986. *Multimind*. Boston: Houghton Mifflin.

Sloman, A. 1987. Motives Mechanisms Emotions. *Cognition and Emotion* 1:217–234.

Sloman, A. 1999. What Sort of Architecture is Required for a Human-like Agent? In *Foundations of Rational Agency*, ed. M. Wooldridge, and A. Rao. : Portland Oregon.

Zadeh, L. A. 1965. Fuzzy sets. *Inf. Control* 8:338–353.

Zhang, Z., D. Dasgupta, and S. Franklin. 1998a. Metacognition in Software Agents using Classifier Systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, Wisconsin: MIT Press.

Zhang, Z., S. Franklin, B. Olde, Y. Wan, and A. Graesser. 1998b. Natural Language Sensing for Autonomous Agents. In *Proceedings of IEEE International Joint Symposia on Intellgence Systems 98*. : .