Metacognition in Software Agents Using Classifier Systems[1]

Zhaohua Zhang
Stan Franklin
Dipankar Dasgupta

Institute for Intelligent Systems
The University of Memphis
Memphis, Tennessee, 38152, USA

Abstract

Software agents "living" and acting in a real world software environment, such as an operating system, a network, or a database system, can carry out many tasks for humans. Metacognition is very important for humans. It guides people to select, evaluate, revise, and abandon cognitive tasks, goals, and strategies. Thus, metacognition plays an important role in human-like software agents. Metacognition includes metacognitive knowledge, metacognitive monitoring, and metacognitive regulation. Conscious Mattie (CMattie), "living" in a Unix machine, automatically reads and understands email concerning seminars (in natural language), and composes and distributes weekly seminar schedule announcements. CMattie implements Baar's global workspace theory of consciousness and some other cognitive theories concerning metacognition, episodic memory, emotions, and learning. Thus, the CMattie project has its cognitive science side (cognitive modeling) as well as its computer science side (intelligent software). This paper describes a case study of the design and implementation of modeling metacognition in software agents like CMattie by using a classifier system.

Keywords: Agent architectures, Genetic algorithms, Software agents, Cognitive modeling, Reinforcement learning

1 Introduction

Software agents are software entities "living" and acting in a real world software environment, such as an operating system, a network, or a database system, and carrying out many tasks for humans. They sense their environment thru their sensors, and act on that environment with their effectors. Their actions effect what they sense in the future, and are in the service of their own drives (Franklin and Graesser 1997). With the rapid development in computer networks, database systems and operating systems in recent years, developing software agents to manage resources in operating systems and information retrieval in network environments, etc. has drawn much more attention. Thus, the research on agent theories, architectures, mechanisms and languages has become much more important.

[1] Partially supported under NSF SBR-9730314

There is still some debate regarding exactly how to define metacognition (Flavell 1976). However most researchers seem to agree that it should include knowledge of one's own knowledge and cognitive processes, and the ability to actively monitor and consciously regulate them. The concepts of self-monitoring, self-evaluation, self-regulation, self-control, self-instruction, self-consciousness, and meta-attention all belong to metacognition. Metacognition is very important for humans. It guides people to select, evaluate, revise, and abandon cognitive tasks, goals, and strategies (Hacker 1997). Implementing metacognition in software agents can be very exciting and challenging. If we want to build more human-like software agents, we need to build metacognition into them. By doing this, we provide agents a meta-system that allows them to overcome internal disorders, to choose an efficient strategy, and to self-regulate.

Conscious Mattie (CMattie) is the successor of Virtual Mattie (Franklin et al. 1996, Zhang et al. 1998, Song and Franklin forthcoming). Virtual Mattie[2] (VMattie) is a less intelligent clerical agent with the same domain as CMattie. CMattie's name derives from the fact that she implements the global workspace theory of consciousness (Baars 1988 1997), along with some other cognitive theories concerning metacognition, episodic memory, emotion, learning, etc. Baar's global workspace theory is a cognitive model of the human conscious experience. CMattie is expected to be more intelligent, more flexible and more adaptive than VMattie. Several functional modules are being added to improve her performance. CMattie's architecture and mechanisms make her "think" and act more like humans do. This paper focuses on a case study of building metacognition into CMattie.

CMattie's brain consists of two parts, the A-brain and the B-brain (Minsky 1985). The A-brain performs all cognitive activities. 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 the A-brain. The B-brain performs all metacognitive activities, and its environment is the A-brain, that is, the A-brain's activities. Figure 1 depicts an overview of CMattie's architecture. In this paper, we will discuss only

[2] VMattie is currently being Beta-tested.

the mechanism of the B-brain and the interaction between some relevant modules in the A-brain and the B-brain. We describe a case study of the design and implementation of metacognition using a classifier system. This system allows the B-brain to satisfy one of the meta-drives of the B-brain, "Stopping any endless loop in which the A-brain finds itself." The endless loop here means that the A-brain repeats itself in an oscillatory fashion. In particular, the B-brain monitors the understanding process of the A-brain, and acts when any oscillation problem occurs. The classifier system allows the B-brain to monitor, to act, and to learn a correct action to stop an endless loop in the A-brain.

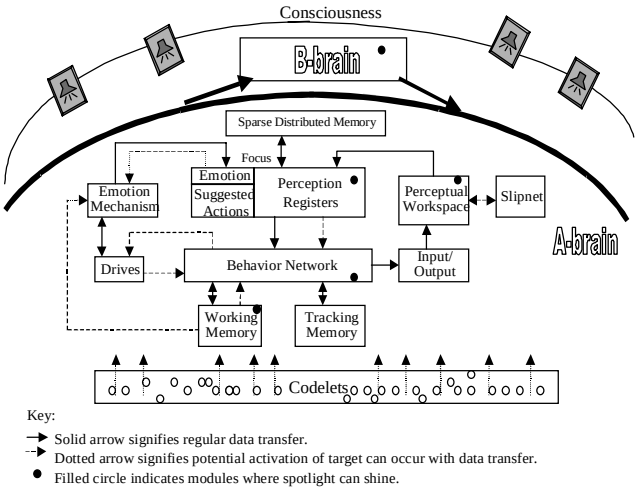

Figure 1: Overview of CMattie's Architecture

2 Problems

Metacognition plays an important role in reading comprehension in humans. Metacognitive monitoring and regulation increase the degree of one's understanding (Ortero 1997). CMattie's reading comprehension consists of understanding incoming email messages concerning seminars in natural language. Metecognition directs CMattie to a higher degree of understanding in that she can handle oscillation problems during her understanding phase.

At present, CMattie's natural language understanding occurs in two stages (Zhang et al. 1998). First, the incoming message is classified as one of the nine message types. This job is done with the help of the slipnet (Hofstadter and Mitchell 1994), an associative memory (see Figure 1 and Figure 3). The nine message types are: Initiate a seminar, Speaker topic, Cancel a seminar session, Conclude a seminar, Change of time, Change of place, Change of topic, Add to mailing list, and Remove from mailing list. For a given incoming message, the nine message-type nodes in the slipnet will have different activation levels. The one with the highest activation level is selected as the proposed message type, a "winner takes all" strategy. But all the other message-type nodes retain their activations and are candidates for the next selection, if the current winner proves to be wrong. The appropriate template is then chosen based on the message type, and placed in the perceptual workspace (see Figure 1). Each message type corresponds to one template. Different message types have different slots in their templates. Figure 2 shows the Speaker-Topic Message template. Codelets (processors) then begin to fill the slots (e.g. speaker name, title of the talk, time of the seminar, date, place, email address of sender, etc.) in the template. If any mandatory slots (e.g. speaker name) are finally not filled, the chosen template, and therefore the proposed message type, is not correct. So the message type with the next highest activation level is chosen as the new proposed message type, the corresponding template is chosen, and its slots are filled. The process repeats until there is a proposed message type with all the mandatory template slots filled. This proposed message type is correct and so is the information in the template. The A-brain performs all the above activities. But if CMattie's A-brain tries to understand an irrelevant message and does not realize that she does not have enough knowledge to do so, the B-brain takes over. It detects the situation and does something to prevent her from repeatedly looking for a message type. For example, a one-node loop could continually circle around a single message type. A classifier system can act as the action selection mechanism for the B-brain. In this particular case, the B-brain monitors whether there is an oscillatory thinking (endless loop) during the A-brain's understanding process, and learns how much activation to send to the nine message-type nodes in the slipnet so that the endless loop is stopped. Figure 3 depicts how the B-brain interacts with the A-brain during the understanding process.

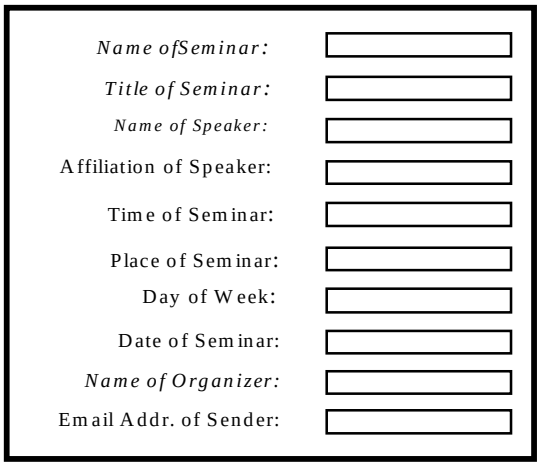

Figure 2: Speaker-topic Message Template
(Italic shows mandatory slots)

3 Mechanism of the B-brain

A classifier system (Holland and Reitman, 1978) is an adaptive system that learns to cope with an environment. Condition-action rules are coded as fixed-length string

rules (classifiers) and can be evolved using genetic algorithms (Holland 1975). The Classifier System in the B-brain is composed of several modules: Inner Perception, Encoder, Message List, Classifier Store, Decoder, Evaluator, Genetic Algorithm and Inner Actions (see the B-brain in figure 3).

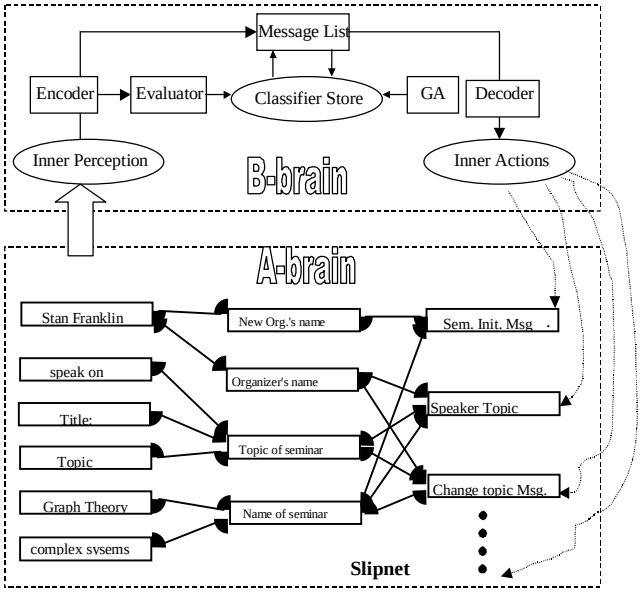

Figure 3: Interaction between the A-brain and the B-brain during the Understanding Process

The Inner Perception Module implements metacognitive monitoring. It consists of sensors and detectors. Detectors differ from sensors in that they do some inferences. Sensors get the raw data from the A-brain, and detectors put them into internal representations. For example, sensors provide information about the current proposed message type or trial template during the understanding process in the A-brain, and detectors check whether there is a loop in the understanding process. So perception can be conceptualized as sensation plus inference. In our case, an inner percept could be of a three-node loop, or a nine-node loop, etc. The inner perception is the B-brain's internal representation of the current state of the A-brain.

An inner percept is then fed to **the Encoder** of the classifier system. The Encoder will encode it to a finite-length string. The B-brain can have ten percepts in binary representation. No loop is encoded as 0000, a one-node loop as 0001, a two-node loop as 0010, etc.[3] After an inner percept is encoded as a string percept, it is put in the **Message List**. This string percept is actually the environment message (or the current state of the A-brain sensed by the B-brain).

The classifier store contains a population of classifiers. Each classifier consists of a condition, an action and a strength, such as 0100 : 001011011110100110, 3.3333. The condition part is a template capable of matching internal string percepts, 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001. The action part consists of sending activation to each message-type node in the A-brain. There are four different levels of activation: low (00), medium low (01), medium high (10), high (11). So the length of an action string is 2*9=18. (nine message-type nodes, each needing 2 bits to represent four levels of activation). The strength serves as a performance measure of the classifier, and ranges from 0 to 9. Since the lengths of the condition part and action part are 4 and 18 respectively, the total number of possible classifiers, ignoring strength, is 2^{22}. The above classifier is interpreted as "if the percept is a four-node loop, then send low activation to initiate-seminar node, medium high activation to speaker-topic node, etc., and this classifier has strength 3.3333."

An initial classifier population of thirty is randomly generated. The strength of each individual is assigned a single value, say 0.5. No domain-specific heuristic knowledge is used to "prime" the initial population. Notice that the strength of a classifier is not the actual performance measure at first. In the beginning, the B-brain has no idea about which rule is good, or which rule is bad. After it takes some actions on the A-brain, and gets feedback (**The Evaluator** changes the strength of a fired classifier), it will have a better idea. The B-brain gradually learns good rules, in other words, a correct action taken on the A-brain in some situations.

At each time step only one classifier acts on the A-brain. Only this classifier is evaluated and its strength updated. All the other unfired classifiers keep their current strengths. New classifiers produced by crossover take the average of their parents' strengths. If the population is too large, the chance of each classifier being fired is low. Convergence will slow down. Experiments show that populations of size over forty are slow to converge, and populations of size less than twenty take longer to stop a loop since there are fewer possible structures. Thirty proved a suitable size for the population and is used by this system.

Once a classifier's condition is matched to the current inner percept, that classifier becomes a candidate to post its action to the **Message List**. It is not possible to let all matched classifiers post their actions there. The length of the message list in this system is ten. The probability of a matched classifier posting its action in the message list is proportional to its strength. The action on the message list that acts on the A-brain is selected at random. A classifier with a high strength does not mean its action is correct. It only means this action is close to the right action. When a correct action is performed, the classifier system will stop since the loop is stopped. On the other hand, some classifiers have high strength because they make the loop smaller. However, we should not give them advantage over others because they cannot stop the loop. If we choose the one with the highest strength every time, some

[3] Binary strings 1010 to 1111 are not used by the system. The classifiers with these condition parts will be deleted by the system.

classifiers with better actions may not have a chance to be fired, and a chance to be evaluated. In the classifier system, only when a classifier is fired and its action is performed, it is evaluated. Randomly selecting an action from the message list gives every active classifier a chance to perform its action and to be evaluated. If no classifiers' condition matches a percept, then some classifiers with lower strengths are selected, and their condition parts changed to match the current percept.

A selected string action is decoded by **the Decoder.** As discussed earlier, 00 is decoded as low, 01 medium low, 10 medium high, and 11 high. Later, **the Inner Actions Module** sends activation to the message-type nodes in the A-brain. The actual activation levels are 0.5 (low), 1.5 (medium low), 2.5 (medium high) and 3.5 (high). The Evaluator decides whether the action provided by a fired classifier is good or bad.

The Evaluator is implemented by a reinforcement learning algorithm (Barto, Sutton, and Brouwer, 1981). It assigns reward or punishment to classifiers based on the next inner percept sensed from the A-brain. Notice that the B-brain has no teacher. In order to see how good or how bad its current action is, it has to see what the next percept is. If after an action is taken, the loop in the A-brain becomes smaller than before, this action gets some reward. If the loop in the A-brain is stopped, this action is a correct action and the classifier system stops.

The bucket brigade algorithm (Holland and Reitman, 1978) is not used in this situation since it is for distributing credit among classifiers. In this system, a good action or a bad action results from a single classifier in each sense-select-cycle. There is no need to distribute reward or punishment.

A sense-select-act cycle is a cycle during which the B-brain senses from the A-brain, selects an action (provided by a fired classifier), and performs the action on the A-brain. However, if the B-brain cannot stop a loop in the A-brain in twenty sense-select-act cycles, the **Genetic Algorithm Module** is activated to evolve new, possibly better classifiers. Classifier's strength is used as a fitness measure.

Genetic algorithms are search algorithms based on natural evolution. In this system, the selection is proportional to each classifier's strength. Only two classifiers with the same condition can participate in a crossover. This allows searching for new actions for a given percept (condition part). Suppose for a given situation in the A-brain, no current classifier has a correct action, crossover may generate a new and correct action to deal with such a situation. The crossover position (point) for each pair of classifiers is randomly generated. The strength of the offspring is the average of its parents' strengths. The rates of crossover and mutation are 1.0 and 0.2 respectively in this system.

In addition to crossover and mutation, this classifier system produces new classifiers using probability vectors (Baluja, Sukthankar, and Hancock, 1997). A probability vector is used to maintain statistics about the correct action string. One probability vector serves all the classifiers with a given condition. There are eighteen numbers in each probability vector since there are eighteen bits in the action part of the classifier. For example, the probability vector for condition 0100 may start as: <0.5, 0.5, 0.5, 0.5, …….0.5>. This means that, in the beginning, the B-brain has no idea about the correct action string. It could be 0 or 1 in each bit position with the same probability. After a classifier 0100 : 011000101100111010 is fired and an action 011000101100111010 acts on the A-brain, suppose the Evaluator gives a middle reward to this action. The probability vector will be updated to close to 011000101100111010 since this action got a reward. It could be updated as <0.25, 0.75,0.75, 0.25, … >. This means the first bit of the correct action string would be more like 0, and second bit 1, etc. Later, if a classifier 0100 : 111001101110111010 is fired and gets a punishment, the probability vector will be updated in the opposite direction. The formula used to update a probability vector is as follows: (Let LR represent the learning rate and i the position in a vector or a string)

Pvector[i] = Pvector[i]*(1-LR) + WinnerVector[i]*LR,
when the winner gets a reward.

Pvector[i] = Pvector[i]*(1-LR) - WinnerVector[i]*LR,
when the winner gets a punishment.

In this way, the B-brain takes every opportunity to learn the probability vector, and keeps a record of such learning. The B-brain updates its probability vector whenever an action is taken. Thus the new classifiers produced by using probability vectors are more likely to be correct.

The system keeps nine different probability vectors, one for each of the nine conditions. When a new classifier is generated from a probability vector, its condition is then associated with the vector. Its action has a 1 in a particular location with the probability found in that location in the vector. The strength of the new classifier is the average strength of the population.

In most GA-based applications, every individual in the population is evaluated at every time step (or generation). In a classifier system, only one individual is chosen and evaluated. So the B-brain must take every opportunity to learn from the feedback of each action. The probability vectors are very helpful in keeping track of what a right action should be. They help the B-brain to learn quickly.

To keep the population at a constant size, new classifiers replace similar members with lower strengths (De Jong 1975).

4 Experimental Results

The system is implemented in JAVA on a Unix workstation. During twenty test runs[4] of the classifier system, on average, the B-brain learned to stop an

[4] Each run used a message which could make CMattie in an oscillatory thinking state.

oscillatory thinking in the A-brain at the 70th sense-select-act-cycle. The fastest case was at the 37th cycle, and the slowest at the 279th cycle. The average clock time for each run is about two minutes.

By an environment, in this context, we mean a particular situation in which an oscillatory thinking (endless loop) is going on in the A-brain during the understanding process. The nine message-type nodes in the slipnet have certain activation levels. The B-brain tries to stop the endless loop by sending different levels of activation to the message-type nodes. In different environments, the A-brain reacts differently to the inner actions from the B-brain. Since the B-brain evaluates classifiers based on any change of the looping state in the A-brain, the A-brain indirectly provides reinforcement to the B-brain. Working in an environment allows the B-brain to find a classifier to stop the endless loop in this particular situation.

The system was tested in two different environments. In both environments, the B-brain successfully learned a that could stop the endless loop classifier in about the 70th sense-select-act cycle.

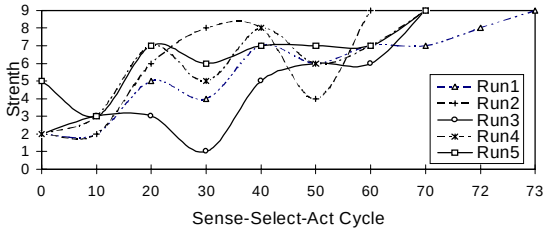

Figure 4: Winner Strenth on Five Runs (in Environment1)
Winner is the fired classifer at each sense-select-act cycle and its action is selected to act on the A-brain.
Its strength reprents its fitness (performance)

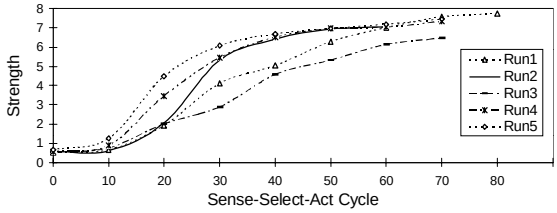

Figure 5: Average Strength of the population on Five Runs (in Environment 1)

We chose five runs from environment 1 and three runs from environment 2 to illustrate the winning classifiers' strength and the average strength of the population at different cycles (shown in Figures 4, 5, 6, and 7). As discussed earlier, at each sense-select-act cycle, one classifier is chosen as the winner and its action is taken on the A-brain. Later it is evaluated and assigned a strength (fitness). In both environment 1 and environment 2, we found that the average strength of the population increased with more sense-select-act cycles. This indicates that the average performance of individual classifiers in the population increased. The overall trend of the winning classifier's strength also increased with more sense-select-act cycles, but sometimes it went down for a while before finally going up.

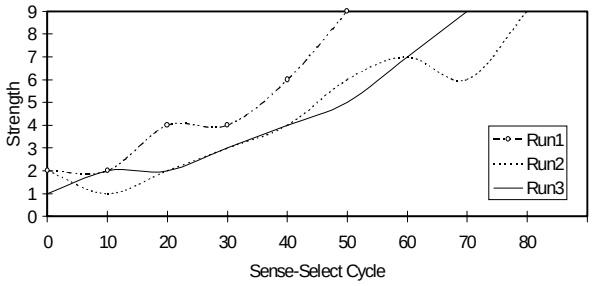

Figure 6: Winner Strength on Three Runs (in Environment2)

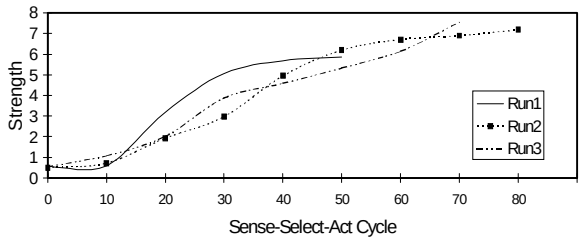

Figure 7: Average Strength of the Population on Three Runs (in Environment 2)

Table 1 and Table 2 show the learned correct classifiers that stopped the endless loops in the A-brain on five different runs in environment 1 and three different runs in environment 2. They also show the sense-select-cycle at which the right actions are performed on each run.

Table 1: Learned Classifiers on Five Runs (in Environment 1)

	Cycle	Learned Correct Classifier
Run1	73	0001 : 000000000000000000
Run2	57	0101 : 000000000000000000
Run3	67	0011 : 000000000000000000
Run4	70	0001 : 000000000000000000
Run5	69	0010 : 000000000000000000

Table 2: Learned Classifier on Three Runs (in Environment 2)

	Cycle	Learned Correct Classifier
Run1	45	0011 : 010000000000000000
Run2	76	0001 : 010000000000000000
Run3	69	0101 : 010000000000000000

In environment 1, the correct action is that the B-brain should send very low activation (0.5) to every message-type-node in the slipnet in any looping state (one-node loop, etc) so as to get rid of the endless loop. For example, on run 1, at 73rd sense-select-act cycle, the B-brain learned a correct classifier: 0001: 000000000000000000.

This means if there is a one-node loop in the A-brain, then send 0.5 activation to all the message-type nodes in the slipnet.

In environment 2, the correct action is that the B-brain should send medium low (1.5) activation to seminar-initial-message-type node and very low activation (0.5) to the other eight message-type-nodes in the slipnet in any looping states. If all the message types have low activation (below certain threshold), the A-brain can realize that the incoming email is an irrelevant message and won't oscillate.

5 Conclusions and Future Work

In this paper, we have used a classifier system to implement the B-brain in order to solve the oscillatory thinking problem in the A-brain. From this case study, we can draw the following conclusions:

- The classifier system proved to be a powerful on-line learning control mechanism, that learns and adapts to its environment.
- A classifier system may be a suitable mechanism for learning metacognitive knowledge by metacognitive monitoring and regulation.
- Classifier systems may converge more slowly than other GA-based applications since only one action is performed and evaluated at each sense-select-act cycle.
- A good way to scale a classifier system is to make it a multiple classifier system so that it distributes several learning tasks to several individual classifier systems.
- Using probability vectors is an efficient approach to speeding up learning in a classifier system.

In future work we will scale this system up to monitor and regulate more activities in the A-brain. We also want to explore a fuzzy classifier system and compare it with a classifier system.

References

Baluja, Sukthankar, and Hancock (1997) Prototyping Intelligent Vehicle Modules Using Evolutionary Algorithms, in Dasgupta, D. and Michalewica, Z. (Eds.) *Evolutionary Algorithms in Engineering Applications*, Springer-Veriag, 1997, pp 241-257.

Baars, Bernard, J. (1988) *A Cognitive Theory of Consciousness*, Cambridge: Cambridge University Press.

Baars, Bernard, J. (1997) *In the Theater of Consciousness*, Oxford: Oxford University Press, Inc.

Barto, A.G., Sutton, R. S., and Brouwer, P. S. (1981). Associative Search Network: a Reinforcement Learning Associative Memory, *Biological Cybernetics*, 40(3): 201-211.

De Jong, K.A. (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral dissertation, University of Michigan.

Flavell, John, H. (1976) Metacognitive Aspects of Problem Solving, In L.B. Resnick (Ed.), *The Nature of In Intelligence*. Hillsdale, NJ: Erlbaum.

Franklin, Stan (1995), *Artificial Minds*, Cambridge: MA: MIT Press.

Franklin, Stan, Art Graesser, Brent Olde, Hongjun Song, and Aregahegn Negatu (1996) Virtual Mattie—an Intelligent Clerical Agent, *AAAI Symposium on Embodied Cognition and Action*, Cambridge MA.

Franklin, Stan and Art Graesser (1997), Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, *Intelligent Agents III*, Springer-Verlag, 21-35

Gilber, A.H. and Bell, F. (1995), Adaptive Learning of Process Control and Profit Optimization Using a Classifier System, *Evolutionary Computation* 3(2): 177-198, MIT Press.

Goldberg, David, E., (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Longman, Inc.

Hacker, Douglas, (1997), Metacognitive: Definitions and Empirical Founditions, In Hacker, D., Dunlosky, J., Graesser A. (Eds.) *Metacogniton in Educational Theory and Practice.* Hillsdale, NJ: Erlbaum, in press.

Hofstadter, D. R. and M. Mitchell, (1994), The Copycat Project: A model of mental fluidity and analogh-making. In Holyoak, K.J. & Barnden, J. A. (Eds.) *Advances in Connectionist and Neural Computation Theory*, Vol. 2: Analogical connections. Norwood, NJ: Ablex.

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

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

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

Minsky, Marvin (1985), *Society of Mind*, New York: Simon and Schuster.

Ortero, Jose (1997) Influence of Knowledge Activation and Context on Comprehension Monitoring of Science Texts, In Hacker, D., Dunlosky, J., Graesser A. (Eds.) *Metacogniton in Educational Theory and Practice.* Hillsdale, NJ: Erlbaum, in press.

Sloman, Aaron (1996) What Sort of Architecture is Required for a Human-like Agent?, *Cognitive Modeling Workshop*, AAAI96, Portland Oregon.

Song, Hongjun and Stan Franklin (forthcoming), Action Selection Using Behavior Instantiation

Wilson, Stewart W. (1994), ZCS: A Zeroth Level Classifier System, *Evolutionary Computation*, MIT Press.

Zhang, Zhaohua, Stan Franklin, Brent Olde, Art Graesser and Yun Wan (1998), Natural Language Sensing for Autonomous Agents In Proceedings of *International IEEE Joint Symposia on Intelligence and Systems'98.*