Integrated Hybrid Cognitive Architecture for a Virtual Roboscout

Alexei V. Samsonovich[1], Giorgio A. Ascoli[1,2], Kenneth A. De Jong[1,3], and Mark A. Coletti[1]

[1]Krasnow Institute for Advanced Study
[2]Psychology Department
[3]Computer Science Department
George Mason University, Fairfax, VA 22030-4444
asamsono@gmu.edu, ascoli@gmu.edu, kdejong@gmu.edu, mcoletti@lychnobite.org

Abstract

The challenge addressed by this research project is to create a hybrid cognitive architecture that will possess key features of human higher cognition. Our approach is based on the integration of symbolic and connectionist components at the top representational level. The framework of schemas, which is the base of this architecture, is described in the context of integration with neuromorphic cognitive maps in specific learning paradigms. The formalism of schemas is illustrated by several examples set in a virtual Roboscout environment.

Introduction

One of the greatest challenges of our time is to create a hybrid, integrated cognitive architecture that will possess key features of human higher cognition sometimes referred to as the "magic" of human cognition. These key features include:

(a) autonomous cognitive growth and human-like learning capabilities,
(b) the basic kinds of human memory and attention control,
(c) social meta-cognition and self-awareness,
(d) human-like communicational capabilities, and
(e) emotional intelligence.

In the present work we address the features (a) and (b).

Approach to Integration

The approach pursued in this work is based on our recent proposal for the integration of symbolic and connectionist components at the top representational level (Samsonovich and De Jong 2005). On the symbolic side of this integration scheme, the key elements are a unique, central notion of a "self" and a formal representation system based on the innovative building block called a "schema", which is the primary focus of the present work. On the connectionist side, the key elements are neuromorphic cognitive map components: neural networks that provide for associative indexing of symbolic memories, path-finding in modeled cognitive spaces (Samsonovich and De Jong 2005), reinforcement learning, and other functions some of which are discussed below. The integration is achieved through an associative learning mechanism between the symbolic and neuromorphic components.

Architecture

In summary, the architecture has eight components that are highly interconnected to each other (Samsonovich and De Jong 2005): working memory, episodic memory, semantic memory, input-output buffer (all these four components operate using symbolic representations: schemas and their instances, see below), procedural memory (interface and "foreign" functions called here *primitives*), a driving engine (an "operating system" that runs all components), a reward and punishment system (used in reinforcement learning and as the origin of goal-directed behavior), and finally a neuromorphic component implementing cognitive maps.

Mental State Lattice

As a part of implementation of the *self concept*, working memory in our architecture is dynamically partitioned into mental states. Each mental state corresponds to an instance of a "self" and represents the mental perspective of that instance, which is partially captured by a self-explanatory mental state label: e.g., I-Now, I-Previous, I-Imagined. None of the mental states are permanent architectural components: mental states may emerge and disappear, move across memory systems, change their labels, functional roles and relationships; however, a certain set of standard labels are likely to be present at any given time in an awake system. Among them are I-Now and I-Next: they represent states of awareness of the subject at present and at the next moment. These and other mental states form a dynamical graph on the lattice of all possible *mental perspectives*. The content of each mental state is a set of instances of schemas bound to the corresponding mental

perspective and to each other. The underlying schema-oriented representation formalism that was described previously at an abstract level (Samsonovich and De Jong 2005) is specifically designed to enable cognitive growth capabilities and to allow for indexing, retrieval, and organization of memories by the connectionist component. This formalism is addressed in the following section.

Cognitive Map

The neuromorphic cognitive map component includes: (1) a *contextual map* that indexes episodic memories, (2) a *conceptual map* that indexes semantic memories, and (3) a separate map for *emotional* memories that may involve episodes as well as semantic knowledge. The notion of a (spatial) cognitive map was introduced by O'Keefe and Nadel (1978). *A cognitive map in the present framework is understood as an abstract multi-dimensional space that is used to index memories.* It can be implemented as a (subset of) the phase space of a neural network (e.g., an attractor).

Therefore, points of the map correspond to dynamical states of the network (patterns of activity). The map can be linked to symbolic memory traces via associative learning.

Virtual Roboscout Paradigm

One possible scenario in which the architecture can be used is the ontogenesis of an agent embedded in a virtual urban scout environment (snapshot shown in Figure 3 A). This general paradigm can be called *"a Car World"*.

The embodied agent is born as a car in a virtual city. It has innate senses, effectors, cognitive abilities and values. The ontogeny of the agent is based on a sequence of scenarios that can be viewed as a specially designed "training facility". These scenarios will be designed to allow the agent to incrementally acquire new cognitive abilities. Some of them may require interactions with an instructor or with other cars controlled by similar virtual agents or human participants. Success in each new scenario will depend on the ability to use previously acquired knowledge. With this general setup, several test scenarios and metrics can be proposed that should specifically address the higher cognitive dimensions (a) and (b) listed above: the "magic" of human cognition.

Schema Formalism

We previously introduced our notion of a schema (Samsonovich and De Jong 2005) as an abstract template used to instantiate mental categories. At that level, our notion of a schema has so many analogies and synonyms that it is impossible to list them all here. The top ten examples would be: classes in an object-oriented programming framework, UML models (Booch, Rumbaugh, and Jacobson, 2005), S-expressions in LISP, frames (Minsky 1975, 1985), productions and operators in Soar (Laird et al. 1986, 1987; Newell 1990), chunks and productions in ACT-R (Anderson and Lebiere 1998), semantic networks, finite state automata, mental models (Johnson-Laird 1983), "specialists" in polyscheme (Trafton et al. 2005). The idea of our approach is fusion and generalization of the above.

On the other hand, the term "schema" originally introduced by Immanuel Kant in the XVIII century is currently used in many fields of science, with dozens of different semantics, some of which appear to be close to our understanding of this term (e.g., Barsalou 1999).

We previously pointed to several distinguishing features of our notion of a schema, one of which is its universal versatility. A schema can be used to represent virtually anything, from an arbitrary set of mental categories (including abstract and concrete concepts, feelings, qualia, actions, volitions, etc.) to a notion as specific as a particular object taken at one particular moment of time. Similar to classes in an object-oriented framework, our schemas give rise to their *instances* (also called *cognitive states*); schemas may be composed of other schemas; more specific schemas can descend from more general schemas, inheriting some of their features; and most importantly, *arbitrary new schemas can be created by schemas*.

All symbolic components of our architecture are based on the formalism of schemas (Figure 1). We pointed out previously (Samsonovich and De Jong 2005) that our notion of a schema allows for a graphical representation. Indeed, a schema in our framework can be viewed as a graph structure composed of nodes that are connected to each other. It remains therefore to explain what are the nodes and the links of the graph, and how do they evolve over time. The best way to clarify these details is to consider specific examples of schemas in action.

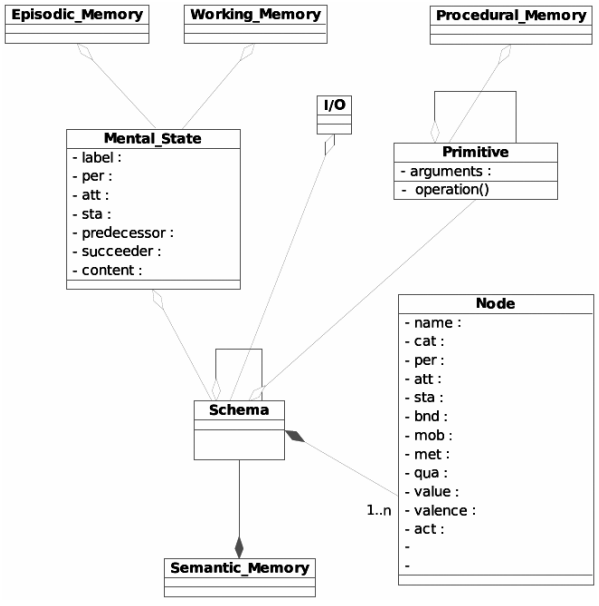

Figure 1. Essential UML class diagram of the symbolic core of the architecture. All nodes of schemas have the standard list of attributes; this is also true about mental states.

First Example: SVD

Bearing in mind the Car World paradigm (see above), we start our consideration with the schema that we call the schema of a simple visible domain (SVD) applicable to 2-D environments. This example is a convenient starting point for us to develop an understanding of the key concepts of this framework. Intuitively, given a global 2-D Cartesian coordinate system, SVD is understood as a rectangular domain defined by two points (the South-West Corner, SWC, and the North-East Corner, NEC). By definition, whenever an agent is located inside SVD, it can see any other object located inside the same SVD. Therefore, the schema of SVD can be written as a list of four elements that we are going to explain below:

(SVD, SWC, NEC, (Obj)*) (1)

In this representation, each of the elements: SVD, SWC, NEC, Obj is a name of a node. A *node* is an atomic token that has a set of standard *attributes* (Figure 1) some of which we describe below. One of the attributes is the *name* of the node: it is a string of characters used for labeling the node. One can think of it as the symbol representing the node. While the choice of the symbol may be important for the human intuitive reasoning, it is actually irrelevant to machine processing of nodes, schemas and instances. Strictly speaking, the node semantics should be inferred not from its name, but from another attribute called Category (*cat*), which has the value of a pointer to a (set of) node(s) of the *global semantic network of categories*, which is a part of the semantic memory system and is used to organize semantic memory.

The first node in (1), SVD, has the name of the schema. This is a consequence of a general rule: the first node of a schema is called the *head* and represents the schema itself. Its name is the name of the schema (or the name of an instance of the schema, when it appears in the instance). The category of the head is the category of the schema. In this framework, there is a one-to-one correspondence between schemas and categories. I.e., each schema has a unique category, and each category is associated with its unique schema. At the same time, the set of all nodes used in all schemas could be much broader. In particular, the category of a node could be any subset of the set of categories indexed by the semantic network.

As long as the semantic network of categories can be viewed as a hierarchy, this hierarchy will entail certain inheritance relationships among schemas, according to the analogy between classes in an object-oriented framework and schemas in this framework. Intuitively one may expect this hierarchy to be organized similarly to the human system of knowledge, starting from the most general categories (entity, property, relation, event, quantity, etc.) and going down to specific instances and atomic elements in the last instance. On the contrary, a given system of schemas (or, the *epistemic state* of the agent) does not have to be complete or even close to completeness.

In addition to schemas that constitute semantic memory, our framework has *primitives* that constitute procedural memory. Accordingly, there are two kinds of nodes that may appear in schemas: those that refer to schemas and those that refer to primitives. The main difference between schemas and primitives is in their implementation. While schemas are all implemented in one universal format (as graphs of nodes, where each node has the standard set of attributes) and are transparent (i.e., all their internal parts are in principle "visible" and accessible for other schemas and instances), primitives in general are opaque ("black boxes") and allow for a proprietary implementation. As far as the protocol of binding is concerned, however, there is no substantial difference between schemas and primitives referenced by nodes. Therefore, primitives are analogous to "foreign functions" in the language based on schemas. While in most cases a schema would be as good a performer as a primitive (and therefore is favorable due to its transparency and versatility), in some special cases the usage of primitives is inevitable. Examples are the input-output channels: sensory signal-to-symbol transduction and the control of effectors and actuators. In addition, in some special "internal reasoning" cases, while schemas in principle could be used, primitives provide a much more rational and powerful choice (built-in arithmetic and logic operations, standard routines like sorting, etc.).

The fact that there are nodes SWC, NEC, and Obj in (1) implies that there are corresponding categories (and the corresponding schemas) available in semantic memory (here we should be more precise: e.g., instead of SWC and NEC, semantic memory may have just one schema of a point; on the other hand, the node Obj may be a template that matches any of several available schemas of objects: e.g., an agent and a gas station, while semantic memory may lack a general schema of a physical object).

How does the SVD schema work? For example, in one possible scenario, our agent may have tunnel vision: it can look at only one location at any moment of time. Furthermore, the agent becomes aware (i.e., reflects in I-Now) of only a subset of all that it can see. Suppose the agent looks at some location *X*, and then traces down to some location *Y*, while seeing nothing special during this focus of attention shift (no objects or walls). Then the agent is aware of two visible locations *X* and *Y*. For an intelligent agent, it would be natural to *assume* in this case that the two points *X* and *Y* belong to the same SVD. Instead, our rather simplistic agent will take this belief for granted. Here is how this happens. When the agent is aware of *X* and *Y*, its driving engine attempts to map the SVD schema onto the content of awareness, and succeeds in binding *X* and *Y* to SWC and NEC, respectively (both nodes SWC and NEC have their mode of binding, or *mob* attribute, set to "Find", which means that under normal conditions of this schema mapping, the binding can only occur when the matching content is found in the target mental state; in addition, by a general rule of binding, unless specified otherwise, the two nodes should match two different entities found in I-Now). A general rule is that the head of a schema should be bound last. In this case, however, the head SVD can be bound as soon as

SWC and NEC are bound, because the node Obj is quantified and allows for an empty set of matching entities. Therefore, the driving engine binds the head of SVD: this completes the first stage of processing of the instance. At a next stage (that will be repeated as long as the instance is not terminated and the mental state remains in the position of I-Now), the driving engine will look for new visible objects that could be incorporated into this SVD.

Returning to the definition of the SVD schema, we now want to modify it by adding a couple of constraints. The first one is a condition that we need just for convenience: the point SWC must be more southern and more western than the point NEC. For now we assume that this constraint is automatically verified by a primitive called "Ordered". A node in our modified SVD schema that will invoke this primitive will be called "Ordered1". In order for this constraint to work, internal links must go from Ordered1 to SWC and to NEC, thereby providing the primitive with access to the coordinates associated with SWC and NEC. Next, we use exactly the same "trick" to make sure that an object associated with SVD is located between the two corners (we assume that the primitive Ordered can take three pairs of coordinates). Now our SVD schema looks like a graph (Figure 2).

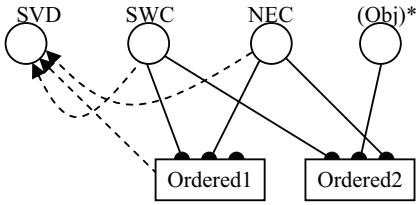

Figure 2. The elaborated SVD schema.

Here the solid lines show internal bindings (all efferent bindings are specified by assigning a pointer to the node attribute called *bnd*), while the dashed arrows indicate that the successful binding of the nodes SWC, NEC and Ordered1 must precede the binding of the head. Again, the second instance of "Ordered" in Figure 2 makes sure that any bound object is located inside the rectangle.

The last terminal node in Figure 2 is not only optional: it is expandable. It is a quantified node that allows for binding a possibly empty list of objects: this fact is represented by the asterisk. There are two possibilities to implement this feature, and our formalism allows for both. One is to get the single node bind to a list of external entities (i.e., the value of *bnd* is a list of pointers). Another possibility is to allow for a duplication of the node, whenever a new object inside SVD is found. This process would be naturally accompanied by the automatic duplication of the internal node Ordered2 together with the related links. The second method, while being more complicated and not obviously necessary in this case, is conceptually important, as it demonstrates the power of the formalism: e.g., a schema may give rise to an instance that grows and becomes more complex than the schema from which it originated (this, however, is not an example of learning or cognitive growth; see below).

Second Example: SM

SVD is a perceptual schema: it detects a certain abstract feature in the environment, marks it (causing by itself no side effects), and stays in awareness for as long as nobody or nothing takes care of its termination. This could mean "forever" (for as long as the simulated cognitive system exists), because I-Now together with its content eventually gets frozen and stored in episodic memory, where normally it should not be deleted or altered. Now we consider another kind of schema: a schema of a self-initiated action of the agent. Action schemas have at least two stages of processing: after being bound, they wait to be executed. We consider a move along a straight line that happens instantaneously for the agent, and we call this schema a *straight move* (SM): Table 1.

Table 1. The SM schema.

name	SM	Me1	Target	Me2	Move1
cat	SM,vact	me	Point	me	move-pr
att	Idea	Nil	Desired	Next	-
mob	Append	Find	Find	Append	-

In order to explain how this schema works, we need to introduce one more attribute called Attitude (*att*). Attitude is understood here as the position (in time and in other cognitive dimensions) of the object of awareness with respect to the subject of awareness (or vice versa – in the intuitive sense: the attitude of the subject toward the object). E.g., if I imagine myself at the next moment of time, the attitude of my imagined body will be "Next" in my I-Now. If I think of a (real) location in space where I want to be, the attitude of my instance of awareness of that location is "Desired", and so on.

Here is an example of how this attribute works. When applied to the content of awareness in I-Now, at the first step, the SM schema (Table 1) finds (*mob*=Find) the body of the agent (Me1: *cat*=me), which has to be the actual, present body: not an imaginary instance of it (this is specified by *att*=Nil). At the same time, it finds the place where the agent wants to be (Target: *att*=Desired). At this point, the instance is bound, has the attitude "Idea" (i.e., a feasible voluntary action), and awaits its execution (according to the voluntary action procedure that we are not going to discuss here, but need to mention that before execution, the attitude must be changed to "Intent"). When intended for execution, the instance (Table 1) creates an imaginary body of the agent me2, which is in fact the expectation of the agent of where it is going to be after the move. This expectation is projected into the neighboring mental state I-Next. The actual, physical move is performed by a primitive *move-pr* (Table 1). When this happens, mental states change their perspectives (and their labels), as previously described in our related publications.

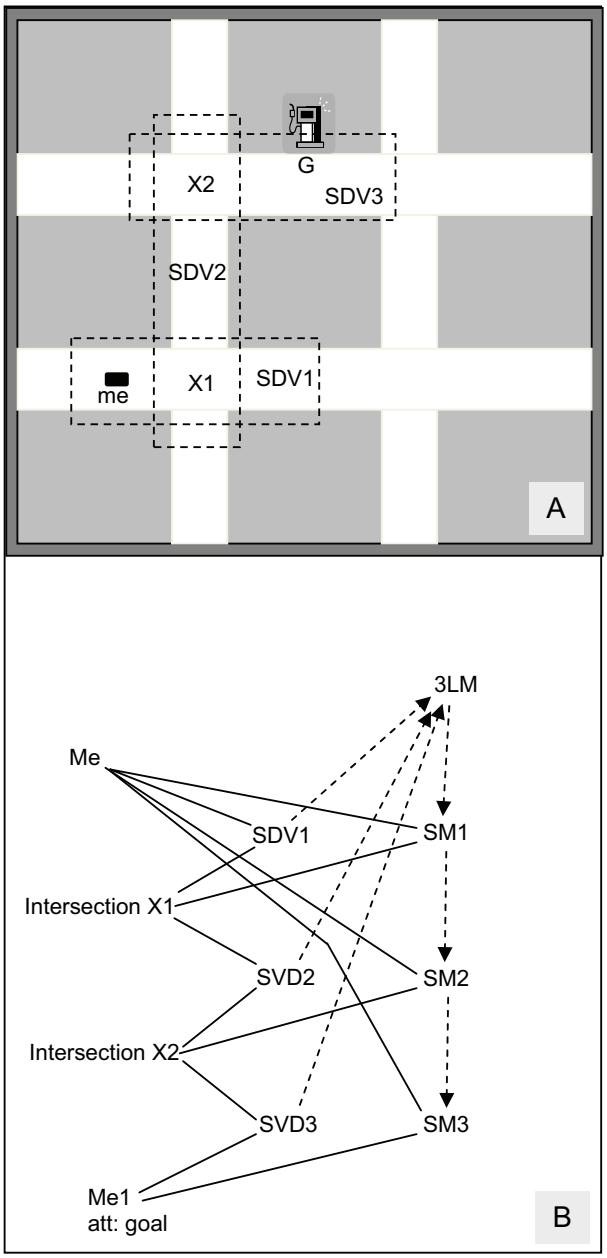

Figure 3. A: navigation to a gas station in a virtual city. B: The schema 3LM (see text).

Third Example: 3LM

Now, jumping ahead, we consider a schema of a three-leg move that we call 3LM (Figure 3 B). It is composed of elements described above, plus one new element: the schema of an intersection. The agent detects intersections in Car World using SVD during exploration of the virtual city. We assume that at this moment the agent has the map of the city available (which could be based on a special map schema). Without going into details in this example, we point that the schema 3LM (Figure 3 B) allows the agent to navigate to the gas station (Figure 3 A): this can be done automatically as soon as 3LM is mapped, which requires solving the matching problem by the driving engine.

Integration with Cognitive Map

What happens in the last example, if 3LM is not available as a schema? The same solution (Figure 3 B) could be found by chance, by exhaustive search, or following guidance of an instructor. Suppose that this happens several times, in several slightly different situations. Variations may include presence or absence of objects in the scene, alteration of the starting and the target locations, alteration of the part of the virtual city where the episode takes place, etc. Each experience (the content of the mental state) is stored in the episodic memory, so that later the system has access to all remembered episodes and related mental state contents in which the solution was found.

Now we assume that the symbolic content of I-Now is linked to an auto-associative neural network that is capable of pattern learning and pattern completion. Details are the following. Each schema is associated with a unique, sparse binary code, which is presented to the neural network as a part of the input pattern, whenever an instance of the schema is present in the working memory. As a consequence, a given mental content in I-Now is represented by an activity pattern in the neural network. We assume that the network is capable of unsupervised pattern learning, generalization (development of a limited number of prototypes for a large number of input patterns), retrieval and completion (e.g., elimination of elements that are not a part of the prototype). Under these assumptions, one may expect the network to develop a generalizing pattern-prototype for a number of presented episodes of successful reasoning of the kind shown in Figure 3 B.

The output of the network in this scenario will be fed to the semantic memory, where it will activate schemas with the binary code that matches the pattern. Schema activation is done via changing the value of the special attribute called “Activation”. More active schemas are more likely to be selected next for an attempted binding to I-Now. Therefore, this mechanism provides the neural network with a means for “suggesting” a missing element of the solution that needs to be added to I-Now. In other words, the neural network is capable of guiding the process of reasoning in I-Now.

Here is why in this paradigm one would need not just any auto-associative network, but a network implementing a contextual cognitive map for successful guidance. The cognitive system may have many experiences, most of which may not be related to the given learning task. Therefore, the system must be able to treat those experiences separately based on their context. The paradigm of finding a path consisting of three legs is a part of the context, and so is a success in this task. With the help of a contextual cognitive map, the neuromorphic

component of the architecture will be able to "filter" the relevant episodes and to generalize them separately from others (details of this process will be presented elsewhere).

Our bottom line here is that the set of instances involved in the solution in Figure 3 B can be associatively learned by a neural network implementing the contextual cognitive map, and this network can be used to "suggest" missing elements of a solution in a new situation based on pattern completion.

A further modification of this paradigm would be to extend the integration described above to the process of automated new schema creation, which is an example of human-like learning and an important component in the cognitive growth. After many experiences of successful reasoning of the sort of Figure 3 B, the system decides to create a new schema (the 3LM schema) by "chunking" the pattern of instances and their connections involved in the solution of the problem. How should our architecture decide which instances to include in the new schema? By means of two criteria: (i) an instance must be productive, in the sense that it should be functionally involved into the solution process; and (ii) it should be recognized by the neuromorphic component as a part of the general prototype pattern for the given class of episodes. Including the neuromorphic component that implements the cognitive maps addresses some of the "chunking problems" that are characteristic of production systems based on symbolic representations only.

Conclusions

In this work we have elaborated new details of the formalism of schemas that underlies our concept of a biologically-inspired cognitive architecture (Samsonovich and De Jong 2005). We presented three examples of schemas, explaining how they solve specific cognitive problems in a virtual Roboscout setup. We explained the indexing and the pattern completion functions of the neuromorphic cognitive map applied to schema-based symbolic representations in working memory. We further explained how the pattern completion function of the cognitive map can be used

(1) to guide the process of "thinking" in the architecture, e.g., to guide the selection of new schemas to be applied to the current mental content in order to solve a problem, and

(2) to facilitate the process of cognitive growth (in this case, new schema creation) by suggesting which instances to include into the new schema.

In general, the functionality of cognitive maps in this framework extends far beyond these two examples; however, these examples provide good illustrations of the new concept of integration of symbolic and connectionist components. Building intelligent systems that exhibit the "magic" of human cognition is a difficult, but probably the most interesting and critical challenge of our time. There are many views of this challenge, and many suggested approaches to its solution. In this paper one particular approach is presented that takes us one step closer to a robust, integrated computational cognitive framework.

Acknowledgments

We are grateful to Mr. Robert Lakatos for critical comments and discussions of the presented material. This work is supported by the DARPA IPTO BICA Grant "An Integrated Self-Aware Cognitive Architecture".

References

Anderson, J. R., and Lebiere, C. 1998. *The Atomic Components of Thought.* Mahwah, NJ: Lawrence Erlbaum Associates.

Barsalou, L. W. 1999. Perceptual symbol systems. *Behavior and Brain Sciences* 22(4):577-609.

Booch, G., Rumbaugh, and J, Jacobson, I. 2005. *The Unified Modeling Language User Guide: Second Edition.* Upper Saddle River, NJ: Addison-Wesley.

Johnson-Laird, P. N. 1983. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness.* Cambridge, MA: Harvard University Press.

Laird, J. C., Newell, A., and Rosenbloom, P. 1987. Soar: an architecture for general intelligence. *Artificial Intelligence* 33:1-64.

Laird, J. E., Rosenbloom, P. S., and Newell, A. 1986. *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies.* Boston, MA: Kluwer.

Minsky, M. 1975. A framework for representing knowledge. In P. H. Winston, ed. *Psychology of Computer Vision,* 211-277. New York: McGraw-Hill.

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

Newell, A. 1990. *Unified Theories of Cognition.* Cambridge, MA: Harvard University Press.

Nichols, S., and Stich, S. 2003. *Mindreading: An Intergrated Account of Pretence, Self-Awareness, and Understanding Other Minds.* Oxford: Oxford UP.

O'Keefe, J., and Nadel, L. 1978. *The Hippocampus as a Cognitive Map.* New York: Clarendon.

Samsonovich, A. V., and De Jong, K. A. 2005. Designing a self-aware neuromorphic hybrid. In Thorisson, K. R., Vilhjalmsson, H., and Marsela, S., eds. *AAAI-05 Workshop on Modular Construction of Human-Like Intelligence: AAAI Technical Report*, vol. WS-05-08, 71-78. Menlo Park, CA: AAAI Press.

Trafton, J. G., Cassimatis, N. L., Bugajska, M. D., Brock, D. P., Mintz, F. E., and Schultz, A. C. 2005. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions On Systems Man And Cybernetics Part A-Systems And Humans* 35(4): 460-470.