Two Paradigms Are Better Than One, And Multiple Paradigms Are Even Better

Arun K. Majumdar and John F. Sowa

VivoMind Intelligence, Inc.

Abstract. During the past half century, the field of artificial intelligence has developed a large number of theories, paradigms, technologies, and tools. Many AI systems are based on one dominant paradigm with a few subsidiary modules for handling exceptions or special cases. Some systems are built from components that perform different tasks, but each component is based on a single paradigm. Since people freely switch from one method of thinking or reasoning to another, some cognitive scientists believe that the ability to integrate multiple methods of reasoning is key to human-like flexibility. In his book *The Society of Mind*, Minsky (1986) presented an architecture for intelligence based on a society of heterogeneous agents that use different reasoning methods to solve different problems or different aspects of the same problem. That idea is intriguing, but it raises many serious issues: how to coordinate multiple agents, distribute tasks among them, evaluate their results, encourage agents that consistently produce good results, inhibit agents that produce misleading, irrelevant, or unfruitful results, and integrate all the results into a coherent response. The most difficult problem is to enable multiple heterogeneous agents, acting independently, to produce the effect of a single mind with a unified personality that can pursue and accomplish coherent goals. This article discusses ways of organizing a society of heterogeneous agents as an integrated system with flexible methods of reasoning, learning, and language processing.

This is a slightly revised version of a paper that was published in the proceedings of ICCS 2009, edited by S. Rudolph, F. Dau, and S.O. Kuznetsov, LNAI 5662, Springer, pp. 32–47, 2009.

1. Architectures for Intelligent Systems

In the years since its founding conference in 1956, the field of artificial intelligence has generated an impressive collection of valuable components, but no comparably successful architecture for assembling them into intelligent systems. The following list illustrates the range of AI components that were designed and implemented in the 1950s and ’60s:

> Parsers, theorem provers, inference engines, search engines, learning programs, classification tools, statistical tools, neural networks, pattern matchers, analogy finders, problem solvers, planning systems, game-playing programs, question-answering systems, dialog managers, machine-translation systems, knowledge acquisition tools, modeling tools, and robot guidance systems.

During the past forty years, all these systems have spawned extensions, combinations, and variations. A recent handbook covered two dozen systems of logic and knowledge representation, each with multiple versions of techniques that tend to be mutually exclusive (Harmelen et al. 2008). Various AI systems use different techniques, but few, if any, take advantage of the full range of options.

Most large systems are designed around a single paradigm, such as formal deduction, statistical language processing, or case-based reasoning. As an example, the largest knowledge-based system, Cyc (Lenat 1995), has millions of axioms, grouped in several thousand contexts or *microtheories*. To

solve different kinds of problems, Cyc uses a few dozen specialized inference engines, but all of them are based on some form of deduction. Partisans of different paradigms have debated their virtues as if they were mutually exclusive, yet most of them have complementary strengths and weaknesses. There should be some way to take advantage of the best features of any or all of them when appropriate.

One way to support divergent methods within a common framework is to partition them among independent processes that run in separate modules. Any such partition would require some way to control the modules and transfer information among them. The fields of AI, computer science, and automata theory have developed several techniques:

> Lambda calculus, abstract machines, subroutines, coroutines, object-oriented protocols, message passing, associative blackboards, Petri nets, π-calculus.

Of these, message passing is the most general method for information transfer, and π-calculus (Milner 1999) is the most general theory for combining control and message passing. Petri nets, for example, can represent single-threaded flow charts, the parallelism of coroutines, object-oriented protocols, and a wide range of asynchronous control mechanisms. Milner showed that π-calculus can simulate the mechanisms of both Petri nets and lambda calculus. But π-calculus goes beyond the fixed graphs of Petri nets by allowing new links to be dynamically created and destroyed. The Linda method of passing messages and control through associatively accessed blackboards (Gelernter 1985) can support π-calculus by its ability to create and destroy links.

The Flexible Modular Framework™ (FMF) proposed by Sowa (2002, 2004) is an architecture for intelligent systems inspired by *The Society of Mind* (Minsky 1986), the Elephant 2000 language (McCarthy 1989), and the message-passing protocols of computer science. As in Minsky's society, each FMF module is an autonomous agent that communicates with other agents by passing messages. As in McCarthy's Elephant, messages can be expressed in logic, but a marker for the speech act indicates the sender's intention. An agent that knows a recipient's identity can send it a message directly, but an agent can find new recipients that can handle a certain kind of message by posting it to a Linda blackboard. The FMF message format has six fields:

1. **Language**. An identifier of the language used in the message. It could be a natural language, a version of logic, or any computer-oriented format.
2. **Source**. An identifier of the module or agent that sent the message.
3. **Message ID**. An identifier generated by the sender.
4. **Destination**. An identifier of the intended receiver, if known. For messages sent to an associative blackboard, this field is null; any module that responds to the message would create a new link.
5. **Pragmatics**. An identifier of the purpose or speech act: command, question, response, assertion, reminder, agreement, concession, estimate, diagnosis, request, promise, contract, etc.
6. **Message**. Any sentence or list of sentences in the language specified by field #1.

Most message formats include most of these fields. The two characteristic features of the FMF are the null option in field #4 for an associative blackboard and the speech act in field #5, which supports an open-ended variety of interaction modes. Without those fields, the FMF can support useful subsets, such as dataflow graphs or Petri nets. With associative blackboards, the FMF can dynamically create and destroy links among agents. With speech acts, FMF agents can express a wider range of intentions than an ordinary command or query. These two fields enable the agents to discover and take advantage of an expanding and evolving range of services created by the system. They also enable agents to look for alternatives if their familiar collaborators are unable to solve an unusual kind of problem.

These formats enable the FMF to accommodate arbitrary modules, even legacy systems, by enclosing them in a wrapper that maps their inputs and outputs to FMF messages. Several variations of the FMF have been implemented, and they use a lightweight protocol that can be implemented in 8K bytes per agent. Thousands of agents can run simultaneously on a laptop computer, but they can communicate with other agents anywhere across the Internet. The messages to and from any user interface have the same six fields as all other messages in the FMF. Therefore, any user interface can be replaced, revised, or enhanced dynamically just by rerouting the messages to a different module. A version of the FMF can be implemented in any language that supports communication among multithreaded processes. At VivoMind Intelligence, Inc., several versions of the FMF were implemented in Java and a multithreaded version of Prolog. But an FMF module can send messages across the Internet to FMF modules implemented in any combination of hardware and software.

Experience in implementing and using FMF systems has shown that an architecture based on message passing among heterogeneous agents has several advantages over more conventional implementations: flexibility of adding new modules without disrupting operations by the old modules; reduction or elimination of systemic errors caused by biases in any single algorithm or paradigm; performance advantages of a lightweight protocol that can take advantage of multiple CPUs; and fail-soft redundancy, which allows most of the agents devoted to a function to continue even if one or more of them fail. Section 2 of this paper describes Minsky's proposals for a society of agents and ways of implementing them. Section 3 describes an organization of FMF agents in a managerial hierarchy that presents a unified personality to the external world. Section 4 describes the use of FMF societies for language analysis. The concluding Section 5 relates the multiple paradigms to Peirce's semiotics and the logic of pragmatism.

2. The Society of Mind

Systems of multiple agents have been proposed and implemented since the early days of artificial intelligence. But the problems of organizing multiple autonomous agents, allocating resources among them, getting them to focus on the relevant goals, and integrating many partial contributions into a unified result have been challenging:

- **Pandemonium.** Selfridge (1959) designed a system of agents called *demons*. Each demon could observe aspects of the current situation or workspace, perform some computation, and put its results back into the workspace. In effect, Pandemonium was a parallel forward-chaining reasoner. Its major drawback was that the demons generated large volumes of mostly irrelevant data that overflowed storage. Since then, a great deal of research has been devoted to measures of relevance, methods for motivating agents to produce relevant results, and ways of allocating resources to those that consistently produce the best results.
- **Rational agents.** At the opposite extreme from simple demons are rational agents that simulate a human-like level of beliefs, desires, intentions, and the ability to reason about them. Van der Hoek and Wooldridge (2008) surveyed versions of logic designed to represent groups or coalitions of such agents. Such logics may be useful for analyzing or simulating the behavior of a group of intelligent agents. But a system with human-like intelligence requires heterogeneous modules specialized for different functions, not a coalition of reasoners that all use the same logic.
- **Reactive agents.** For designing robots, Brooks (1991) noted that the major challenge was not in deliberative planning and reasoning, but in the seemingly simpler insect-like functions of perception, locomotion, and goal seeking. That observation stimulated work on reactive agents

> whose intelligence is at the level of ants. A society of such agents can cooperate in defending the colony, searching for food, and caring for the eggs and larvae. But no one has shown how a colony of ants could understand language or do complex reasoning and planning.

Complex rational agents and simpler reactive agents operate at different extremes of intelligence. But most systems consist of one kind or the other, not a combination of heterogeneous agents. After many years of examining different ways of designing and implementing intelligent systems, Minsky (1986) argued that no single mechanism, by itself, can adequately support the full range of functions required for a human level of intelligence:

> What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle. Our species has evolved many effective although imperfect methods, and each of us individually develops more on our own. Eventually, very few of our actions and decisions come to depend on any single mechanism. Instead, they emerge from conflicts and negotiations among societies of processes that constantly challenge one another. (Section 30.8)

In a review and critique of AI systems, Minsky (1991) emphasized that each of the many paradigms had made valuable contributions, but that the goal of a homogeneous system built around a single, ideal paradigm was too narrow to support the full range of human intelligence:

> The functions performed by the brain are the products of the work of thousands of different, specialized sub-systems, the intricate product of hundreds of millions of years of biological evolution. We cannot hope to understand such an organization by emulating the techniques of those particle physicists who search for the simplest possible unifying conceptions. Constructing a mind is simply a different kind of problem — of how to synthesize organizational systems that can support a large enough diversity of different schemes, yet enable them to work together to exploit one another's abilities.

In an earlier paper, Minsky (1980) proposed an administrative organization populated by "mental managers" that employ and direct other agents that perform tasks at varying levels of complexity:

> To develop this idea, we will imagine first that this Mental Society works much like any human administrative organization. On the largest scale are gross "Divisions" that specialize in such areas as sensory processing, language, long-range planning, and so forth. Within each Division are multitudes of subspecialists — call them "agents" — that embody smaller elements of an individual's knowledge, skills, and methods. No single one of these little agents knows very much by itself, but each recognizes certain configurations of a few associates and responds by altering its state.

As an example of the diversity of modules, Figure 1 shows the interconnections among the kinds of modules proposed by linguists. The large box at the bottom would contain a much larger collection of modules for all the aspects of cognition and behavior that provide the subject matter and the goals for language and reasoning.

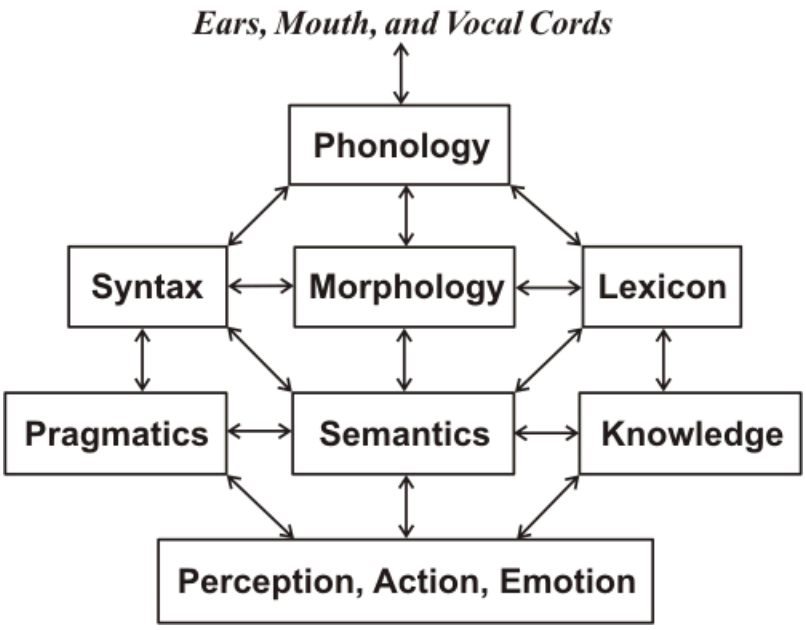

Figure 1. Interconnections among language modules

The diversity of modules that process language is a subset of the even greater diversity in all aspects of cognition and behavior. The integration of language with every aspect of human perception, behavior, and social interaction suggests that the language modules are interconnected with other cognitive modules in dynamically changing ways. Whatever the organization, the number of modules is undoubtedly far greater than the eight boxes of Figure 1. Perhaps there is no limit to the number of modules, and every language game and mode of behavior has its own module or even a supermodule composed of multiple smaller modules. That organization is radically different from a homogeneous system based on a logic that cannot tolerate a single inconsistency. Minsky's goal was to build a flexible, fault-tolerant system out of imperfect, fallible components. Such a system could support logic, just as the flexible, fault-tolerant, and fallible human brain supports logic, mathematics, and every branch of science, business, and the arts. More recently, Minsky (2006) emphasized the role of emotions in driving an engine composed of multiple agents. Without emotions to set the goals, a logic-based theorem prover would have no reason to do anything.

As the underlying mechanism for implementing agents, Minsky continued his long-term research on neural networks. His newer proposals are based on knowledge lines or K-lines that pass information and control to activate agents or even a cascade of agents. In a review of Minsky's theories, Singh (2003) compared the Society of Mind to the SOAR architecture for a "unified theory of cognition" (Newell 1990):

> To the developers of SOAR, the interesting question is what are the least set of basic mechanisms needed to support the widest range of cognitive processes. The opposing argument of the Society of Mind theory is that the space of cognitive processes is so broad that no particular set of mechanisms has any special advantage; there will always be some things that are easy to implement in your cognitive architecture and other things that are hard. Perhaps the question we should be asking is not so much how do you unify all of AI into one cognitive architecture, but rather, how do you get several cognitive architectures to work together?

That question is the central theme of Minsky's book, but Singh concluded that the complexity of the ideas and the lack of detail has discouraged implementers: "While SOAR has seen a series of

implementations, the Society of Mind theory has not. Minsky chose to discuss many aspects of the theory but left many of the details for others to fill in. This, however, has been slow to happen."

The lack of detail plagues many proposed models of the mind. In the book *What is Thinking?* Baum (2004) surveys attempts to simulate thinking and includes a dozen citations to Minsky's *Society of Mind*. Following Minsky, he assumes "the computation of the mind is rich, with modules connected to modules, flowing in complex flow patterns" (p. 35). Baum views Minsky's mental managers and administrative organizations as participants in an economy guided by Adam Smith's "invisible hand" (p. 241):

> The agents in the economy will be computer programs, initially random computer programs. They will be rewarded by the economy, and the ones that go broke will be removed. New entrepreneurs will enter. Hopefully, if we get the economic structure right so that the individuals are rewarded appropriately, the system will evolve to solve hard problems... Now, we want to look at what's going on in an economy regarded as an evolutionary system consisting of a bunch of agents, each evolving to pursue its own interest, each evolving purely to increase its pay-in. We want to ensure that this evolution nonetheless promotes the overall functioning of the whole system.

Starting evolution from random computer programs would take a long time, but using economic rewards as a management tool seems promising. In fact, the economists Monderer et al. (2001) propose game theory for devising reward strategies that could motivate AI agents. A working system, however, requires much more attention to implementation detail.

3. A Hierarchy of Managers and Employees

The modules of the Flexible Modular Framework can be organized in an open-ended number of ways, and various strategies have been implemented and tested. One of the first had a graphic interface that allowed a software designer to drag and drop agents on a screen and connect them in a graph that resembles a Petri net. That was a useful tool for rapidly assembling modules, but it did not have a graphic way of showing the links found by means of associative blackboards. Another application replaced the fixed programs of an interactive game with FMF agents. The game graphics and the types of characters and machines were unchanged, but the FMF agents gave them more flexible ways of interacting, behaving, and communicating. The most general version implemented at VivoMind exploits Minsky's idea of a hierarchy of managers and employees. The chief executive officer (CEO) gives the organization a coherent "personality" for external interactions. Beneath the CEO are vice presidents in charge of major divisions, directors of important functions, lower-level managers, and specialists that perform an open-ended variety of cognitive tasks. As an example, Figure 2 shows the upper levels of a hierarchy designed to analyze and interpret natural language texts.

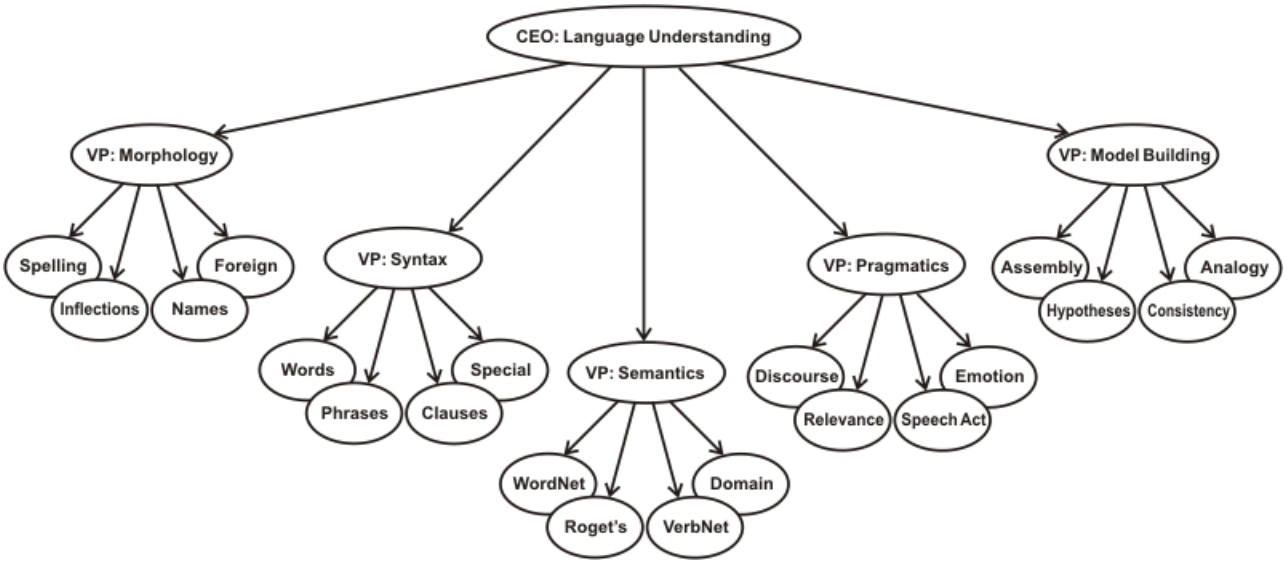

Figure 2. A management hierarchy of language processing agents

At the top of Figure 2, the CEO of language understanding is responsible for all functions from the analysis of individual words (morphology) to the construction of a "mental model" of the meaning of an entire text, which could be a sentence, a paragraph, a conversation, a report, or a book. Reporting to the CEO are vice presidents in charge of the divisions of morphology, syntax, semantics, pragmatics, and model building. Beneath the vice presidents are directors in charge of functions such as spelling correction in the morphology division and parsing in the syntax division. The semantics division has directors of domain ontologies for the detailed axioms of the subject matter and directors of lexical resources, such as WordNet, Roget's Thesaurus, and VerbNet. For the current implementation, a formal ontology for the upper levels has not been helpful. Detailed reasoning is done with specialized ontologies for the subject matter, and the lexical resources have been adequate for mappings between English text and the specialized domain ontologies. The hierarchy shown in Figure 2 is a composite that shows the typical functions performed by the agents. Most of the implementations have more levels for middle managers, first-level managers, and specialist employees.

As in Minsky's administrative organizations, management control flows down from the CEO at the top, many messages flow up and down the hierarchy, but messages can also flow sideways across the hierarchy. In a review of the SOAR architecture, Minsky (1993) observed that the chunking mechanism of SOAR corresponds to the production of K-lines in the Society of Mind. For the VivoMind implementations, the basic knowledge representation is conceptual graphs (Sowa 2008). Chunking is implemented by a *lambda abstraction*, which defines a new concept or relation type by a conceptual graph (CG) in which one or more concept nodes are identified as formal parameters. An instance chunk is defined by assigning a name to an entity described by a CG. These mechanisms can encode frequently occurring patterns of graphs in single concept or relation nodes. The names or type labels correspond to K-lines that link all occurrences of that chunk.

Minsky maintained that a system of heterogeneous agents should allow agents to use a multiplicity of languages tailored for their purposes. The language field in an FMF message supports an open-ended variety of languages, but the Conceptual Graph Interchange Format (CGIF) is the *lingua franca* for reasoning and language processing. Two agents implemented in the same language, such as Prolog, can also exchange the equivalent information in their native language format. An untranslated input language can be represented by a concept node whose referent is an uninterpreted character string:

```
[EnglishSentence: "This is an example of an English sentence."]
```

Minsky (1991) claimed that an AI system should support "neat" methods based on formal logics as well as "scruffy" methods based on informal heuristics. With current technology, any translation from an unrestricted natural language is at best a useful, but scruffy approximation. Some FMF applications also use a version of Common Logic Controlled English (CLCE), which has an unambiguous mapping to conceptual graphs whose semantics are defined by the Common Logic standard (ISO/IEC 24707). Anyone who can read English can read a CLCE statement, but some training in logic is necessary to write syntactically correct CLCE. With a clarification dialog, a person who is not a trained CLCE author can work with a help facility to convert an informal English sentence to a CLCE statement that both the human and the computer can accept. For many applications, however, a scruffy translation from ordinary English can be valuable (Majumdar et al. 2008).

Singh (2003) noted the pitfalls of relying on blackboards as the primary method of communication among agents: "While the blackboard metaphor may work when there are only a few agents using the blackboard, by the time there are hundreds of agents, let alone thousands or millions, the image of them huddled around a blackboard is no longer reasonable, and in fact no one has built a blackboard system of this scale." For that reason, most FMF messages are sent to a known recipient, but an FMF system can have an open-ended number of blackboards, which may be used in various ways:

1. **Newsletter**. Any agent that manages other agents may set up a blackboard for notes that members of the department may post to any or all members of the group. The CEO might use global newsletters to announce information that could be accessed by any agents in the hierarchy, or even by unemployed "freelance" agents.
2. **Agenda**. A blackboard may serve as a queue of tasks to be done, and any available agent that can handle a task could remove it from the queue and do it. Some kinds of jobs could be performed by multiple agents, and a manager might let several perform the job and select the best results.
3. **Want ads**. A manager could post a job description to a global blackboard accessed by freelance agents that might offer their services.
4. **Classified advertising**. Freelance agents might offer to sell data or hypotheses on blackboards that are specialized for a variety of purposes.
5. **Committees**. Blackboards used for collaborative reasoning would normally be restricted to a small group of agents that resemble a committee. Such a group would fit the metaphor of collaborators "huddled around a blackboard." Committees form a collaborative environment where agents can evaluate options, vote for their preferences, or negotiate to combine them.

Variations of these five uses for FMF blackboards have been implemented in systems for language and reasoning (Majumdar et al. 2008) and in a game-playing system for knowledge capture (Majumdar et al. 2007). For each agent, one local blackboard is the default. An agent can access other blackboards indirectly by sending a request to an agent for which the desired blackboard is local.

At VivoMind, the authors have developed a technique called *Market-Driven Learning*™ (MDL), which rewards agents with resources: computer space and time to perform their services. A hierarchy that reports to a CEO can earn resources by providing services to external users or systems. The CEO distributes resources as rewards to the vice presidents, who distribute their allotment to the managers that report to them. The managers can use their resources to hire employees, reward employees for good performance, or buy data and hypotheses from freelance agents or from other managers. The managers may combine the data and hypotheses themselves or assign their employees the task of doing the combination. Managers can also serve on committees to negotiate for resources or to produce committee reports to be sent up the hierarchy. Managers at each level of the hierarchy receive rewards

from higher levels, they reward their employees for what they produce, they can hire new employees or fire unproductive employees, and they can buy or sell data and services by sideways transfers to other managers.

An MDL society learns by reorganizing itself to produce improved results, which humans or other agents are willing to buy. The reward system addresses the basic problems faced by Pandemonium: increasing resources for the most productive agents; reducing resources for the less productive agents; and reorganizing the hierarchy by growing the more productive branches and shrinking the less productive branches. The rewards pass through the management hierarchy to create an effect similar to the *backward propagation* learning of a neural network. But unlike the simple switches and numeric functions of a neural network, MDL agents can be arbitrarily complex programs or reasoning systems, they can hire or fire other agents, and the messages can be propositions or even large documents stated in some version of logic. If the messages are stated in a dialect of Common Logic, they could be translated to CLCE in order to provide humanly readable explanations or an "audit trail" about the way the FMF system derived its data, hypotheses, and reports. These options are not possible with the numeric weighting schemes of most neural networks. (Note, however, that individual agents in an FMF system could use any computing mechanism internally, including neural networks. But such agents would communicate with other FMF agents by the usual FMF message formats.)

4. Interpreting Natural Languages

A system that interprets natural language must take into account all the aspects of language covered by the eight boxes in Figure 1. As that diagram suggests, every aspect is related, directly or indirectly, to every other aspect. Psycholinguistic studies indicate that people process all those aspects concurrently, and brain scans indicate that different aspects seem to be processed in different parts of the brain. None of the psychological or neurological studies, however, are sufficiently detailed to show the internal data formats or the kinds of operations performed on that data. As a working hypothesis, many linguists and computational linguists have assumed that the underlying conceptual structures can be conveniently represented by labeled graphs, possibly with nested graphs within graphs. That assumption is very general, since it includes most of the alternatives as special cases: strings, trees, feature structures, and various notations for logic. Conceptual graphs are a semantic representation influenced by the research in linguistics, logic, psycholinguistics, and computational linguistics (Sowa 1984, 2008). They can represent ISO standard Common Logic as a proper subset, but they can also be processed by scruffy heuristics.

For the VivoMind implementations, conceptual graphs are generated in the semantics division in the center of Figure 2, and they are further elaborated in the pragmatics and model-building divisions at the right. For any input text, the morphology and syntax divisions at the left usually begin the processing, but the VP agents that manage the other divisions run concurrently. Therefore, they can begin to make partial contributions to the analysis before the morphology and syntax agents have finished the sentence. As an example, the following sentence appeared in a text about oil and gas exploration:

> The Diana field is situated in the western Gulf of Mexico
> 260 km (160 mi) south of Galveston
> in approximately 1430 m (4700 ft) of water.

If the sentence had ended with the word *Mexico*, the syntax would be unambiguous. But the measures in the next two lines, the parenthetical expressions, and the points for attaching prepositional phrases create ambiguities. Is Diana field or the Gulf of Mexico south of Galveston? What is in the water? Diana field, the Gulf of Mexico, or Galveston? After a devastating hurricane, Galveston was under water, but background knowledge should imply that cities are usually not under water.

Agents that process lexical information, context, heuristics, and domain knowledge contribute to the interpretation. A morphology agent expands "ft" to "feet". An ontology for the geoscience domain indicates that Diana field is a reservoir, which consists of rocks that trap hydrocarbons; such a reservoir is underground; and the ground may be under water. Parenthesized expressions are usually idiosyncratic and ad hoc. One agent detected measures that were approximately equal, but stated in different units. Therefore, it made the hypothesis that the parenthesized expressions were intended to express equality. During the parsing process, the agents can create multiple links as tentative hypotheses. A manager in charge of those agents evaluates the evidence for each alternative and prunes away the unlikely options. The remaining links indicate that Diana field is south of Galveston and in the water.

Many different syntactic parsers have been used to generate conceptual graphs. But theories that focus on the connections between words, such as *dependency grammars* and *link grammars*, are convenient because their links map directly to the nodes and arcs of CGs. Several different parsers have been implemented at VivoMind, and the ones based on link grammar (Sleator & Temperley 1993) have been the easiest to combine with graph operations for semantics. The latest VivoMind parser is based on link grammar, but influenced by a distributed concurrent parser called ParseTalk (Hahn et al. 1994, 2000). Instead of the static global control of conventional parsers, ParseTalk has a "dynamic, local-control model" that supports "a balanced treatment of both declarative and procedural constructs within a single formal framework." The ParseTalk control structure is based on *actors* implemented in an object-oriented language (Smalltalk). Bröker (1999) added semantic actors to the original syntactic actors of ParseTalk. The actors enable it to process multiple syntactic constraints and knowledge sources concurrently. Zeman and Žabokrtský (2005) combined the results of multiple dependency parsers by a committee that used voting and learning. The committee choices were significantly more accurate than the results of the best parser by itself.

The ParseTalk actors and the FMF agents have similar advantages, but the object-oriented actors are more tightly coupled than the heterogeneous FMF agents. As the developers said, ParseTalk has "a single formal framework." For the FMF agents, the only thing that is common to all of them is the message format with six fields. Different agents can use different languages, different paradigms, and even different hardware located on different continents. The loose coupling of the FMF agents makes it easy to add new agents with new capability without disrupting any of the older functions; it also enables the system to continue if some agent or agents fail. In some applications, one or more FMF agents failed, but the system continued to run without their input. Eventually, the manager of the agents restarted them.

5. Reasoning with Multiple Paradigms

Deduction is the most common method of reasoning used with logic-based systems. But deduction is precise, predictable, and brittle. If everything is perfect, deduction is perfect. Such perfection is only achievable in pure mathematics. For normal, imperfect computer applications, deduction can magnify and propagate any imperfection to the point of a total collapse. When people reason, they seldom carry out long chains of deductions, and they often perform a "sanity check" to avoid obvious errors. If a conclusion seems odd, a prudent individual would ask for advice or try an alternative method of reasoning. People don't expect every message to be completely understood. They ask questions, give explanations, negotiate, and compromise. In short, they use multiple paradigms to cross-check their results and avoid the biases that tend to occur with just a single paradigm. Bundy and McNeill (2006) maintain that intelligent systems must have a similar ability to revise, repair, and refine their axioms to accommodate the inevitable exceptions, discrepancies, and aberrations.

Frege and Peirce were pioneers in logic, who independently discovered equivalent representations for full first-order logic. But they had different goals for logic. Frege applied his logic to mathematics, for which deduction is the primary method of reasoning. But Peirce used logic in a much broader range of applications, including scientific discovery, philosophical analysis, and the definition of words in linguistics and lexicography. In addition to deduction, Peirce emphasized the use of *induction* in generalizing from examples and *abduction* in forming hypotheses or educated guesses. Unlike many logicians who viewed metaphors and analogies with suspicion, Peirce (1902) included analogy as one of the four methods of reasoning: "Besides these three types of reasoning there is a fourth, analogy, which combines the characters of the three, yet cannot be adequately represented as composite." Figure 3 is a diagram of Peirce's cycle of reasoning, inspired by his lectures on pragmatism (1903).

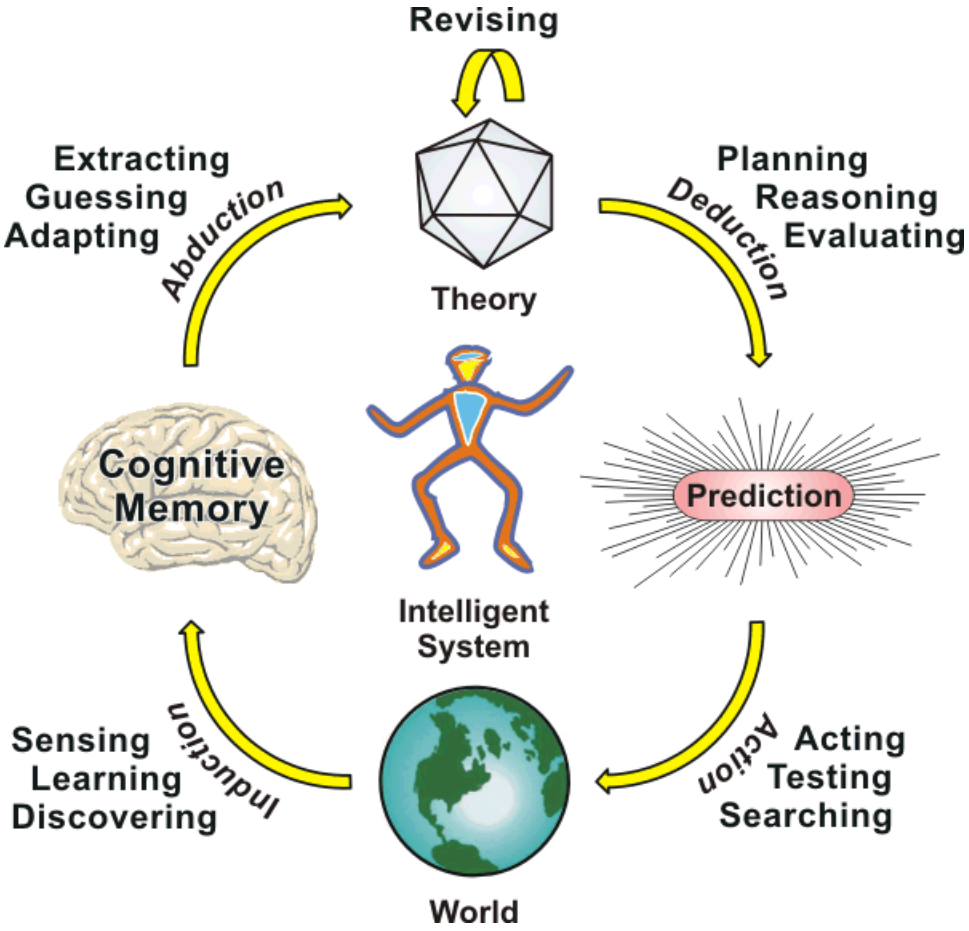

Figure 3. Peirce's cycle of pragmatism

Note that deduction is a small part of the cycle. By itself, deduction can only derive the consequences of already familiar assumptions. Induction forms generalizations from specific instances, abduction supports guessing or hypothesis formation, revision accommodates exceptions, and testing keeps reasoning grounded in reality. Analogy combines aspects of the other methods of reasoning, and it can be used by itself as the primary method for informal reasoning. The brain in Figure 3, labeled cognitive memory, represents an open-ended associative store of all the knowledge and data acquired by a system, natural or artificial. In capital letters, Cognitive Memory™ is a high-performance associative memory developed by VivoMind.

A critical component of any intelligent reasoner is a high-speed associative memory for finding relevant chunks, K-lines, schemata, or other patterns of knowledge. For conceptual graphs, that would

require a high-speed method for indexing and finding relevant graphs and subgraphs. Some of the most advanced research on processing graphs has been done by chemists, who need to classify and search for millions of graphs of organic molecules. An application of chemical algorithms to conceptual graphs led to the first high-speed method for classifying and finding conceptual graphs (Levinson & Ellis 1992); one implementation of that method was used in the web site of a large online retailer (Sarraf & Ellis 2006). More recent work on chemical graphs has produced algorithms for encoding both the graph structure and the labels in numeric vectors, indexing the encodings, and finding all graphs within a small semantic distance of a given query graph (Rhodes et al. 2007); those algorithms are being used to index and search a database of over four million chemical graphs. Those techniques resemble the methods for indexing conceptual graphs and finding analogous graphs in logarithmic time (Sowa & Majumdar 2003):

1. Convert each graph to a unique linear representation. For a chemical graph, the conversion is based on its International Chemical Identifier (InChI). Similar conversions can be applied to labeled graphs of any kind.
2. Map the linear form to numeric vectors that encode both the graph structure and the ontology (labels) on the nodes and arcs.
3. Use a measure of semantic distance between the vectors. For conceptual graphs, that measure takes into account both the structure (ordering, connectivity, and cycles) and the ontology (type labels and hierarchy). For chemical graphs, similar structural properties are used, but the ontology is based on the properties of atoms and chemical bonds.
4. Use the semantic distance measure to index the graphs and find graphs within a given distance (threshold).

For conceptual graphs, the time to build the index is proportional to (N log N), where N is the number of graphs. The time to find graphs that are similar to a given query graph is proportional to (log N). If more than one graph is found within a given threshold, structure-mapping algorithms can be used (Falkenhainer et al. 1989), but it's often faster to distinguish graphs by applying semantic operations directly to the encodings.

The Flexible Modular Framework with multiple heterogeneous agents has proved to be a flexible, robust, and efficient system for learning, reasoning, and language processing. The six-field message format together with associative blackboards supports the computational power of the π-calculus. The Cognitive Memory system provides a high-speed resource for analogy finding, case-based reasoning, and associative access to knowledge and information of any kind — including facts and axioms stated in Common Logic and transmitted in CGIF. The Market Driven Learning methods with the rewards of resources for good performance extend the π-calculus to a version of the $-calculus or cost-calculus by Eberbach et al. (2004). A cost measure based on space and time requirements can constrain the excesses of systems like Pandemonium and direct agents toward promising goals. Turing (1939) showed that a Turing machine that could ask for information from a external *oracle* was more powerful than a Turing machine in isolation. Eberbach et al. claimed that messages from external agents or the environment could serve as the equivalent of an oracle to go beyond the limitations of a Turing machine. Whatever the theoretical power, the FMF with these additions has served as a practical tool for rapidly building intelligent systems.

References

Baum, Eric B. (2004) *What is Thought?* MIT Press, Cambridge, MA.

Bröker, Norbert (1999) *Eine Dependenzgrammatik zur Kopplung heterogener Wissensquellen,* Max Niemeyer Verlag, Tübingen.

Brooks, Rodney A. (1991) Intelligence without representation, *Artificial Intelligence* **47**, 139-159.

Bundy, Alan, & Fiona McNeill (2006) Representation as a fluent: An AI challenge for the next half century, *IEEE Intelligent Systems* **21:3**, pp. 85-87.

Eberbach, Eugene, Dina Goldin, & Peter Wegner (2004) Turing's ideas and models of computation, in C. Teuscher, ed., *Alan Turing: Life and Legacy of a Great Thinker*, Springer, Berlin.

Falkenhainer, B., Kenneth D. Forbus, Dedre Gentner (1989) The structure mapping engine: algorithm and examples, *Artificial Intelligence* **41**, 1-63.

Gelernter, David (1985) Generative communication in Linda, *ACM Transactions on Programming Languages and Systems*, pp. 80-112.

Hahn, Udo, Susanne Schacht, & Norbert Bröker (1994) Concurrent natural language parsing: The ParseTalk model, *International Journal of Human-Computer Studies* **41**, 179-222.

Hahn, Udo, Norbert Bröker, & Peter Neuhaus (2000) Let's ParseTalk: Message-passing protocols for object-oriented parsing, in H. Bunt & A. Nijholt, eds., *Recent Advances in Parsing Technology*, Kluwer, Dordrecht.

ISO/IEC (2007) *Common Logic (CL) — A Framework for a family of Logic-Based Languages*, IS 24707, International Organisation for Standardisation.

Lenat, Douglas B. (1995) Cyc: A large-scale investment in knowledge infrastructure, *Communications of the ACM* **38:11**, 33-38.

Levinson, Robert A., & Gerard Ellis (1992) Multilevel hierarchical retrieval, *Knowledge Based Systems* **5:3**, 233-244.

Majumdar, Arun K., John F. Sowa, & John Stewart (2008) Pursuing the goal of language understanding, in P. Eklund & O. Haemmerlé, eds, *Proceedings of the 16th ICCS*, LNAI 5113, Springer, Berlin, 2008, pp. 21-42.

Majumdar, Arun, Mary Keeler, Paul Tarau, & John Sowa (2007) Semantic distances as knowledge capture constraints, *First Workshop on Knowledge Capture and Constraint Programming (KCCP-2007)*, Whistler, BC.

McCarthy, John (1989) Elephant 2000: A programming language based on speech acts, http://www-formal.stanford.edu/jmc/elephant.html

Milner, Robin (1999) *Communicating and Mobile Systems: The π calculus*, Cambridge University Press, Cambridge.

Minsky, Marvin (1980) K-lines: A Theory of Memory, *Cognitive Science* **4**, 117-133.

Minsky, Marvin (1986) *The Society of Mind*, Simon and Schuster, New York.

Minsky, Marvin (1991) Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy, *AI Magazine*, **12:2**, 34-51.

Minsky, Marvin (1993) Review of Allen Newell's *Unified Theories of Cognition*, in *Artificial Intelligence*, **59**, 343-354.

Minsky, Marvin (2006) *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*, Simon & Schuster, New York.

Monderer, Dov, Moshe Tennenholtz, & Hal Varian (2001) Economics and artificial intelligence, *Games and Economic Behavior* **35:1-2**, 1-5.

Newell, Allen (1990) *Unified Theories of Cognition*, Harvard University Press, Cambridge, MA.

Peirce, Charles S. (1902) *Logic, Considered as Semeiotic*, MS L75, edited by Joseph Ransdell, http://www.cspeirce.com/menu/library/bycsp/l75/l75.htm

Peirce, Charles S. (1903) Harvard lectures on pragmatism, in N. Houser & C. Kloesel, eds., *Essential Peirce*, vol. 2, Indiana University Press, Bloomington, pp. 133-241.

Rhodes, James, Stephen Boyer, Jeffrey Kreulen, Ying Chen, & Patricia Ordonez (2007) Mining patents using molecular similarity search, *Pacific Symposium on Biocomputing* **12**, 304-315.

Sarraf, Qusai, & Gerard Ellis (2006) Business rules in retail: The Tesco.com story, *Business Rules Journal* **7:6**, http://www.brcommunity.com/a2006/n014.html

Selfridge, Oliver G. (1959) Pandemonium: A paradigm for learning, in *The Mechanization of Thought Processes*, NPL Symposium No. 10, Her Majesty's Stationery Office, London, 511-526.

Singh, Push (2003) Examining the society of mind, *Computing and Artificial Intelligence* **22:6**.

Sleator, Daniel, & Davy Temperley (1993) Parsing English with a link grammar, *Third International Workshop on Parsing Technologies*, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/link/pub/www/papers/ps/LG-IWPT93.pdf

Sowa, John F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA.

Sowa, John F. (2002) Architectures for intelligent systems, *IBM Systems Journal* **41:3**, 331-349. http://researchweb.watson.ibm.com/journal/sj41-3.html

Sowa, John F. (2004) Graphics and languages for the Flexible Modular Framework, in K. E. Wolff, H. D. Pfeiffer, & H. S. Delugach (2004) *Conceptual Structures at Work*, Proceedings of ICCS 2004, LNAI 3127, Springer, Berlin, pp. 31-51.

Sowa, John F. (2008) Conceptual graphs, in van Harmelen et al. (2008) pp. 213-237.

Sowa, John F., & Arun K. Majumdar (2003) Analogical reasoning, in A. de Moor, W. Lex, & B. Ganter, eds. (2003) *Conceptual Structures for Knowledge Creation and Communication*, LNAI 2746, Springer, Berlin, pp. 16-36.

Turing, Alan M. (1939) Systems of logic based on ordinals, *Proc. London Mathematical Society*, Series 2, Vol. 45, pp. 161-228.

van der Hoek, Wiebe, & Michael Wooldridge (2008) Multiagent systems, in van Harmelen et al. (2008) pp. 887-928.

van Harmelen, Frank, Vladimir Lifschitz, & Bruce Porter, eds. (2008) *Handbook of Knowledge Representation*, Elsevier, Amsterdam.

Zeman, Daniel, & Zdeněk Žabokrtský (2005) Improving parsing accuracy by combining diverse dependency parsers, *Proc. IWPT-05*, ACL, pp. 171-178.