CHAPTER VIII

SUMMARY AND CONCLUSIONS

8.0 Summary and Conclusions

The methodology of knowledge-based AI relies on a central assumption concerning the nature of intelligence:

> The [knowledge-based AI] paradigm has traditionally assumed that the symbolic, algorithmic character of the macrophenomena [of intelligence] also characterizes the inner workings of the cognitive processor that generates the macrophenomena. (Chandrasekaran, Goel and Allemang 1988, p. 32)

Thus, knowledge-based AI models intelligence only at an abstract level and assumes the resulting models adequately describe the underlying process. At this abstract level, which Newell (1981, 1982) calls the knowledge level, there is only knowledge, actions, goals and a rational application of the knowledge to satisfy the goals with the actions. Each intelligent action realized by a knowledge-based AI program is explainable by some piece of knowledge within the agent that associated the action to a specific context.

The most direct way to implement intelligent behavior then is to create a knowledge-base that embodies the pertinent aspects of the task environment so it can be placed into the problem solver. Once all relevant knowledge is explicit within the problem solver, the task is reduced to deciding what knowledge to apply when. If appropriate knowledge is unavailable, the mechanism can simply fall back to an ordered search until the stored knowledge is again applicable. This is the essential philosophy of weak and strong methods in knowledge-based AI and has been shown time and again to be sufficient for modeling intelligent behavior.

The difficulties associated with knowledge-based AI's straightforward approach to modeling intelligence lie not in the recapitulation of intelligent behavior but in the distinctness between the task environment and the problem solving environment. For the problem solver to have direct access to knowledge about the task environment, the task environment must be modeled within the problem solver. This requires a transduction of information from its natural form in the task environment into an internal format directly accessible by the problem solver. Every nuance of the task environment that is germane to solving the problem must be reconstructed within the agent so it can be invoked at appropriate points. This transduction leads to problems of credit assignment, knowledge acquisition, scaling, knowledge indexing and representation design which are the classic problems of knowledge-based AI. The motivating thesis of this work is that because of the problems associated with explicit knowledge, literal knowledge level models may be an inappropriate level at which to describe a general computational intelligence. A straightforward way to remove these problems is to remove their source, namely the insistence on explicit task knowledge within the problem solver. The objective of this dissertation has been to investigate the necessity of knowledge-based AI's commitment to providing explicitly represented knowledge within an agent to produce intelligent computational activity.

In Chapter 2, I described evolutionary algorithms, and in Chapter 3 I argued that they were indicative of a new weak method that increased its informedness about the organization of the search space dynamically during the search. Evolutionary weak methods manipulate a population of solutions much as weak methods like best-first and beam search. Evolutionary weak methods distinguish themselves by their use of representation-specific operators, their propagation of abstract features and the processing strength of reproduction. These factors allow them to be task independent but still perform more like strong methods than weak methods. Also in Chapter 3, I identified a weak form of credit assignment in these algorithms, termed it empirical credit assignment after its empirical nature, and generalized Holland's (1975) analysis of genetic algorithms to describe the propagation of beneficial abstract features through the population. The distinguishing feature of empirical credit assignment is its ability to adapt its search in a problem independent manner as additional points of the search space are encountered.

In Chapter 3, I used the properties of empirical credit assignment to hypothesize a novel sub-branch of AI techniques. Rather than assuming that all knowledge must reside

within an intelligent agent, it was proposed that the interaction of a simple agent with the environment could also give rise to intelligent behavior without associated knowledge being explicitly represented within the agent or the environment. Thus, the knowledge that determines the intelligent behavior emerges out of the interaction of the simple agent and the environment, a computational theme that goes back at least to Simon (1969) and is the foundation of situated activity (Greeno 1993; Clancey 1993). The techniques proposed in this dissertation extend this work by demonstrating that interactions between a general problem solver and the task environment can also determine task-specific modifications of the problem solver's internal representations. I dubbed this method of AI emergent intelligence and I argued that at the knowledge level, a knowledge-based AI agent and an agent whose knowledge emerges are indistinguishable. Because emergent intelligent systems appear similar to knowledge-based AI methods at the knowledge level but contain no explicit task knowledge, the knowledge in these systems must emerge at the abstract level from the interaction of the simple problem solver with the task environment. Following the identification and characterization of emergent intelligence, the next four chapters demonstrated this concept through experimentation.

Chapter 4 described an experiment where an evolutionary algorithm was able to induce an appropriate neural network architecture for several tasks without task-specific knowledge. I argued that the heuristics typically used by connectionist researchers to induce an architecture are overly simplistic in their approach. GNARL, the evolutionary algorithm used in these experiments, created appropriate architectures for several tasks while inducing the parametric values simultaneously. GNARL was shown to induce complete networks for several language learning tasks in comparable time to a second method that only induced weights for a designed architecture. In these experiments, the generalization of the networks created by GNARL was consistently better. In another experiment, GNARL induced networks to perform a simple control task. These experiments demonstrated that two networks induced by GNARL might solve the same task in much the same manner although their method of achieving the solution differs significantly.

The experiments of Chapter 4 illustrate emergent intelligence at a simple and straightforward level. Appropriate architectures for the networks are induced by repeatedly manipulating them with representation-specific operators and testing them in the actual task environment. When a network architecture appears that provides better problem solving accuracy than the average ability of the current population, it and its variants receive

increasingly more of the population space and hence more subsequent testing. Because there was no task-specific knowledge available within the problem solver to determine an appropriate network architecture, this knowledge emerged at the more abstract knowledge level without explicit instantiation in the environment or the problem solver.

Chapter 5 furthered the experiments of Chapter 4 by adding two representation-specific operators that protected components of the FSAs evolving in the population. The added operators made only syntactic manipulations of the solutions rather than the typical semantic manipulations. The intention was that the evolutionary algorithm would determine which components of the FSA representation were crucial to solving the task and protect them from manipulation by the other reproduction operators. Experimental results showed that given the same experimental conditions, the evolutionary algorithm with the additional operators identified solutions more quickly. Rather than requiring an automated analysis of each FSA to determine which components were crucial to solving the task and thus should be protected from manipulation, the interaction of the evolutionary algorithm and the task environment allowed the pertinent knowledge to emerge dynamically.

In Chapter 6, automatic task decomposition was investigated, without an explicit description of the properties of an appropriate decomposition inside the problem solver. Experiments in this chapter evolved modular LISP programs using GLiB, an evolutionary algorithm that evolves a population of LISP programs, to learn both the legal moves and a competitive strategy for solving the Towers of Hanoi problem and playing Tic-Tac-Toe against a rule-based expert. Again, two additional reproduction operators were introduced into the basic evolutionary algorithm that probabilistically selected and defined functions from the programs in the population using only syntactic modifications to the evolving LISP programs. The appropriateness of each randomly defined function was determined by their subsequent proliferation through the population. Beneficial functions were propagated to many population members while inappropriate modules died out with the programs that used them. The contents of the modules induced by the process appeared to be applicable to a broad number of situations within the limitations of representation and performance. An interesting side effect of these experiments is that induced functions defined task-specific representational abstractions. Each new function, when appropriate for the task, defined a more abstract level of representation above the original language that was tailored to the task. In these experiments, both the applicability of a particular module for

solving the task and the abstraction it embodied emerged from the simple problem solver interacting with the task environment.

The final set of experiments, described in Chapter 7, took a closer look at the role of the fitness function in evolutionary algorithms. Standard objective fitness functions are usually simple evaluation functions. However, the numerical value returned by these functions is often arbitrary and is occasionally a liability when an accurate objective function is not easily provided for the task. In addition, evaluation functions are often designed to provide a specific gradient to ensure the success of the learning process. In the experiments of this chapter, the fitness function used in Chapter 6 was varied for the Tic-Tac-Toe task to observe the difference in quality in the discovered modular LISP programs. The various experts used to evaluate the evolving programs included one that chose its position at random, one that was an optimal strategy for the task and one identical to the expert used in the experiments of Chapter 6. A fourth fitness function that used no expert but instead used the population members to play against themselves in a competitive tournament to determine relative rankings was also tested. This evaluation function merely identified which of the two programs passed to it was the better player. The results showed that the competitive tournament induced LISP programs that were more robust players than each of the fitness functions that used rule-based experts. I argued that this resulted from an ecology of strategies developing in the population which provided a pressure for the evolving LISP programs to generalize their performance over a wider array of strategies. Equivalent generalization was not apparent in the programs evolved using the single, non-deterministic rule-based strategies in the fitness function. Thus, a progression toward the implied goal of optimal Tic-Tac-Toe performance emerged from the interaction between population members in the environment without an explicit gradient supplied in the fitness function.

From the experimental results presented in this dissertation, some features of emergent intelligence as implemented by evolutionary algorithms can be identified. First, the dynamics of evolutionary algorithms create a race condition that favors portions of the search space that are dense with near-solutions. For an offspring to be retained as a parent in the subsequent generation, it must perform the task about as well as an average member of the population. Thus, an offspring's performance must be comparable to its parent. The offspring must also be within the search neighborhood of the parent where the search neighborhood is defined relative to the reproduction operators. For subsequent offspring to

be equally viable, they must again be competitive with the average abilities of the population. Thus, the reproduction operators must have again mapped to a comparable solution within the same search neighborhood. Eventually, a section of the search space begins to fill the population and the search is concentrated in that area. However, it is not necessarily the case that only a single area of the search space is represented in the population. Islands of distinct areas in the search space that are all dense with near-solutions can also develop. The relative amount of space occupied by a particular area of the solution space will be proportional to the density of solutions in the neighborhood. The race then is for representational space in the population and thus for assigned credit and future manipulation. As demonstrated in Chapters 5 and 6, this feature of evolutionary algorithms can be exploited to produce beneficial task-specific syntactic organizations that speed-up acquisition of solutions and determine appropriate task-specific modularizations and high-level abstractions without the use of explicit knowledge.

A second feature of emergent intelligence as implemented by evolutionary algorithms is the opportunistic nature of the acquired structures. In each of the experiments of this dissertation, the structures that evolved displayed little similarity with a human problem solver's preconceptions of how to solve the task. The solutions often took advantage of representational constructions that would be too subtle or too complex for a human to design. This was evident in the experiments with GNARL and the Tower of Hanoi experiment with GLiB. As the representation language became more general, the resulting structures became increasingly exploitative of non-standard representational techniques. For instance, in the Tic-Tac-Toe experiments with GLiB of Chapters 6 and 7, the *<test>* portion of the conditional in the representational language often contained numerous side effects. This invariably lead to representations that were non-normative for the problem and extremely cumbersome to describe. As mentioned previously, these representations appear similar in spirit to the distributed representations of connectionist networks (Hinton, Rumelhart and McClelland 1986; Sejnowski and Rosenberg 1987; Smolensky 1988) which are also constructed using an opportunistic mechanism. This property is likely to be present in all implementations of emergent intelligence.

Thirdly, emergent intelligence relies on the task environment, implemented as the fitness function in evolutionary algorithms, to work as a filter for organizations that are limited relative to the rest of the population. The abstract features and opportunistic structures within the population are not induced for their objective truth but instead for their relative

usefulness in solving the task and their ability to bias future search toward increasingly better solutions. The more useful a structure or feature is in the context of the current population, the more variations of it that will be explored in subsequent iterations of the search. This is distinct from the general optimization approach of some intelligent systems. Rather than finding the optimal construction, these methods merely locate one that is useful, i.e., sufficient. If there are idiosyncracies to the structure within some tolerance, they are worked around. Often even these idiosyncracies turn out to be useful, as shown in the Towers of Hanoi experiment of Chapter 6. Hence these techniques are closer to satisficing systems than function optimizers. If optimality is a concern, it can be a criterion inserted into the fitness function for problems when satisficing is unacceptable.

In the future, studies which further illuminate the differences between knowledge-based AI and emergent intelligence are needed. In this dissertation, I claimed that the ills of knowledge-based AI originated from its commitment to explicit knowledge. I then described and demonstrated a specialization of AI that avoided this commitment and still appeared intelligent. This removes the need for explicit knowledge in the problem solver and hence the associated difficulties. However, I did not demonstrate that emergent intelligence can be used to solve the same problems as knowledge-based AI more quickly or efficiently or that they truly circumvent the classic problems of knowledge-based AI systems. I have no doubt that there are problems for which emergent intelligent techniques are beneficial and others for which it is not. The distinctions between these two classes of problems should provide information about both the limitations of knowledge-based AI methodology and the practice of emergent intelligence. Similarly, it is clear that not all the knowledge necessary to solve every problem will always be adequately accessible through emergent mechanisms. The distinctions between what types of knowledge are accessible and what types are not should also provide significant insight into these techniques and possibly the nature of human intelligence. Other future work should include the benefits and limitations of inserting optimality constraints into the fitness function and the identification of a theory of environmental feedback.

Knowledge-based AI techniques assume that the knowledge level is the appropriate level at which to model intelligent agents. This level of description is certainly convenient for testing a model of intelligence to ensure that it is consistent with itself and the observations that gave rise to it. But the symbolic level favored by knowledge-based AI is not necessarily the best level at which to formulate a computational intelligence. This

straightforward approach to modeling intelligence presents numerous computational difficulties when realistic problems are modeled due to the transduction of information from the task environment to the problem solver. Opportunistic methods that dynamically acquire pertinent task information from the environment provide a more general methodology for constructing intelligent systems effectively and circumvent our biases as designers and observers. Emergent intelligence suggests a class of techniques that avoid the modeling commitments of knowledge-based AI while still appearing "as if" they contain equivalent resources. Clarifying the distinctions between knowledge-based AI systems and emergent intelligent systems that perform equivalently, such as those described in this dissertation, will provide significant insight into the nature of human and computational intelligence, the limitations and benefits of computational modeling, and the biases of our scientific observations.