Preprint version of "An Essay Concerning Human Understanding of Genetic Programming," by Lee Spector, to appear in *Genetic Programming Theory and Practice*, edited by Rick Riolo and Bill Worzel, Kluwer (now Springer), 2003, pages 11-24.

Chapter 2

AN ESSAY CONCERNING HUMAN UNDERSTANDING OF GENETIC PROGRAMMING

Lee Spector
Cognitive Science
Hampshire College
Amherst, MA 01002 USA
lspector@hampshire.edu

Abstract This chapter presents a personal perspective on the relation between theory and practice in genetic programming. It posits that genetic programming practice (including both applications and technique enhancements) is moving toward biology and that it should continue to do so. It suggests as a consequence that future-oriented genetic programming theory (*mathematical* theory, developed to help analyze, understand, and predict system behavior) should also borrow, increasingly, from biology. It presents specific challenges for theory vis-à-vis recent technique enhancements, and briefly discusses possibilities for new forms of theory that will be relevant to the leading edge of genetic programming practice.

Keywords: biology, development, representation, diversification, phylogeography, visualization.

1. Theory and Practice

Researchers in the field of genetic and evolutionary computation borrow features of biological systems to produce adaptive problem-solving technologies. For the earliest such technologies, only the most general, skeletal features of biological genetics (as then understood) were appropriated. For example, many early systems represented candidate solutions as linearly structured, positionally encoded genomes, borrow-

ing loosely from the structure of DNA discovered by Watson and Crick in 1953. In addition, these systems used algorithms for mutation and recombination based loosely on DNA mechanics, and schemes for population-level adaptation based on an abstraction of Darwinian natural selection.

What has the role of *theory* been in this endeavor historically? By "theory" here I mean *mathematical* theory, developed to help analyze, understand, and perhaps predict the behavior of genetic and evolutionary computation systems in quantitative terms. Under this definition theory has been essential in the field from its earliest days; for example, Holland's seminal monograph on genetic algorithms, first published in 1975, was largely theoretical, and introduced the formal notion of "schemata" which continues to drive much of the theory in the field today (Holland, 1992). In much of the early work theory and practice occurred together and directly motivated one another.

Genetic programming, by which I mean the specialization of "genetic and evolutionary computation" to systems that evolve executable computer programs, arose later, in the late 1980's and early 1990's. In the succeeding years, genetic programming theory and practice (in which I include both applications-oriented work and development of new techniques) have both undergone steady progress, although not always in a coordinated ways.

With the basic theory already established, system builders and applications engineers felt free to experiment with variations ("hack"), without paying much attention to mathematical theory. For the theorists, on the other hand, there was much work to be done in clarifying and developing the mathematics, irrespective of the activities of the hackers. As a result, there has been a divergence. Understandably, theory has tended to lag behind practice; it is easier to hack systems than it is to integrate the resulting hacks into a rigorous mathematical framework for the purposes of analysis and prediction.

For example, recent advances in genetic programming schema theory have broadened its applicability to include variable-length and tree-structured genomes, along with popular mutation and crossover algorithms (see, for example, Poli, 2001). This was a substantial and successful mathematical undertaking, but for all of its successes it captures a state of the art that is already about 10 years old. During the intervening decade practice has flowered, producing much more complex and capable systems for which there is not yet a strong theoretical framework.

One example of the gap between theory and practice is *automatically defined functions*, a technique extension developed in 1994 that has since gained widespread acceptance, and is believed by many to be essential

for the application of genetic programming to difficult, large-scale problems (Koza, 1994). There is not yet a mathematical theory of genetic programming that provides any real leverage in understanding how and why automatically defined functions are helpful. Meanwhile, practice has moved beyond "simple" automatically defined functions, to systems with *architecture-altering operations* (Koza, 1995) or other mechanisms that allow the number of automatically defined functions and other aspects of modular architecture to evolve as a system runs (Spector and Robinson, 2002).

Should theoreticians therefore turn their attention to extending schema theory to handle automatically defined functions and architecture-altering operations? I do not think that is likely to be the most productive move. Automatically defined functions and architecture-altering operations are only the tip of the iceberg. Among the other innovations that are playing an increasingly important role in genetic programming practice are developmental processes (at genetic, morphological, and behavioral levels), evolved representations, structured "deme" geography, and coevolution. The prospect of extending current theory to cover these complications seems daunting indeed, an of course the innovations in practice continue to accumulate at a rapid clip.

None of this should be taken to dispute either the utility of early, "simple" genetic programming systems for certain classes of problems, or the utility of the newly enhanced theoretical apparatus for understanding and improving these systems. But practice is moving ahead rapidly, and this may be an opportune time to take stock of the situation and to contemplate the possibilities for new forms of theory that will be relevant to the leading edge of practice.

2. To Life!

Can we characterize the specific directions in which genetic programming practice is moving? This would be a tall order, as genetic programming is a field of rapid and sprawling innovation. The on-line genetic programming bibliography[1] contains $3,068$ entries as of this writing, the great majority of which refer to publications within the last ten years. A high percentage of publications in the field describe innovative enhancements to the basic technique; several collections have a roughly tripartite division into theory, technique enhancements, and applications. Even if only something like 10% of the publications in the last ten years describe innovations in genetic programming practice, and I believe this is a conservative estimate, there have nonetheless been several hundred extensions to the technique since the field first blossomed in the early

1990's. Clearly, a systematic survey and analysis of the "direction" in which all of these innovations are pushing the field would be a major undertaking (and beyond the scope of this chapter).

Nonetheless, one can make some rough generalizations. Innovations in technique are usually made by researchers who find that their systems hit some sort of limit before they can solve the problems put to them. When this happens the researchers often turn to other fields for inspiration.

If the perceived limit is one of raw computational resources (memory or computing time) then researchers often turn to other areas of computer science for solutions, for example by optimizing representations or procedures (as in Keijzer, 1996), modifying the techniques to take better advantage of available hardware and software architectures (Spector and Stoffel, 1996; Nordin, Banzhaf, and Francone, 1999; Poli and Langdon, 1999), or adding features derived from software engineering (e.g. modules, as in Koza, 1994; Spector, 1996).

If, on the other hand, the perceived limit is in the adaptive capacity of the technology, then researchers often turn to the science that studies the best known examples of adaptive systems: biology. For example, recent advances in genetic programming technique use mechanisms derived from DNA dynamics (Koza, 1995; Hansen, 2003), learning mechanisms in neural networks (Teller, 1999; Downing, 2001), immune systems (Nikolaev, Iba, and Slavov, 1999), regulatory networks (Banzhaf, 2003), and biological gene expression processes (Keller and Banzhaf, 2001; Ferreira, 2001; Wu and Garibay, 2002).

As a result, a significant portion of the leading edge of the field continues to head toward biology. Even in work that appears quite distant from biology on the surface, one often finds arguments that aspects of the proposed techniques were motivated by some hither-to underappreciated feature of biological systems. Insofar as the adaptive power of biological systems still vastly outstrips that of any currently existing human-engineered computational system this is quite reasonable, and it is a trend that we should expect to continue.

3. Life Evolving

Biology, however, is a moving target. Conceptions of the fundamental mechanisms of biological evolution have changed repeatedly and dramatically since the 1960s and 1970s, when the core concepts of genetic and evolutionary computation were established. One often-cited example concerns the role of symbiosis and other "mutualisms"; originally thought to have been curious exceptions to ordinary biological processes, these phenomena are now widely thought to have played central roles in

the development of multicellular life (Maynard Smith and Szathmáry, 1999), with some researchers arguing that they are major sources of evolutionary novelty on par with natural selection (Margulis, 2000).

Indeed, the very notion of a "gene," which has suffered dramatic semantic shifts in its 94-year history,[2] now appears to be deeply problematic. Evelyn Fox Keller, after describing several recently discovered complications to the traditional picture of gene expression and gene/protein interaction, writes:

> Techniques and data from sequence analysis have led to the identification not only of split genes but also of repeated genes, overlapping genes, cryptic DNA, antisense transcription, nested genes, and multiple promotors (allowing transcription to be initiated at alternative sites according to variable criteria). All of these variations immeasurably confound the task of defining the gene as a structural unit. (Keller, 2000, p. 67)

So, if we are to continue to move in biological directions, we might expect to see the notion of "gene" that is implicit in our genetic representations to undergo significant alterations. Indeed, we might also expect the boundaries between the "genetic" representations and the other components of our systems to continue to blur. For example, Keller argues that the biological mechanisms that generate variation are themselves under genetic control, and that the evolutionary processes are thereby themselves products of evolution:

> The critical dependence of genetic stability on proofreading and repair enzymes may have come as a great surprise, but more surprising yet was the discovery of "repair" mechanisms that sacrifice fidelity in order to ensure the continuation of the replication process itself—and hence the survival of the cell. Far from reducing error, such mechanisms actively generate variations in nucleotide sequence; moreover, it appears that when and where they come into play is itself under genetic control. (Keller, 2000, p. 32)
>
> That regulation of genetic stability and mutability is a feature of all living systems in now widely accepted. (Keller, 2000, p. 35)
>
> Recent laboratory studies of bacterial evolution provide further confirmation, lending support to the notion that organisms have evolved mechanisms for their own "evolvability." (Keller, 2000, p. 37)

Mechanisms related to these ideas have been used in genetic and evolutionary computation for some time (for example, in *evolution strategies* (Bäck and Schwefel, 1995), and in discussions of the evolution of evolvability (Altenberg, 1994)). The lesson to draw from Keller's observations, and from the expected further "biologicalization" of genetic programming hypothesized in this chapter, is that these sorts of mechanisms are likely to play an increasingly important role in the field. To be relevant to future genetic programming practice, therefore, genetic pro-

gramming theory should be applicable to, and informative with respect to, systems with these sorts of extensions.

4. Challenges

In this section I would like to highlight a few specific challenges for genetic programming theory based on recent practice. These are not meant to be exhaustive, and they are drawn mostly from my own work (for no better reason than the fact that these are the challenges with which I am most familiar). Nonetheless I believe that they can provide useful pointers to opportunities for future-oriented theory.

4.1 Evolved Code Structure

As noted above, genetic programming systems now routinely include mechanisms for the evolution of *automatically defined functions* (Koza, 1994) or *automatically defined macros* (Spector, 1996). Other mechanisms for evolved modularity have also been developed (for example, Kinnear, 1994). Because several studies have shown that these features can significantly extend the reach of genetic programming, useful theory ought to provide firm mathematical leverage with respect to questions about how and when automatic modularization can enhance the adaptive powers of genetic programming systems.

The essential difficulty here is that the semantic structure of a modular program can be related to its syntactic structure in complex ways. For example a smaller program may execute more primitive operations than a larger program, even without considering loops, if the smaller program makes more significant use of modularity. Changes to a program will likely have effects of a magnitude that is correlated not only with the "depth" at which the changes occur, but also with the relation between the changes and the program's modular architecture; for example a change within module will probably have an impact that is proportional to the number of times that the module is called.

The challenge is more severe, but also more necessary, for systems in which the "architecture" of evolving programs is itself subject to variation and natural selection. By "architecture" I mean the number of modules and the nature of their interfaces to each other and to the "main program" (or the "result-producing branch" in Koza's terminology). Koza's system of architecture-altering operations employs new genetic operators (in addition to standard crossover and mutation) to permit the addition and deletion of modules, along with other architectural changes (Koza, 1995). These new genetic operators were explicitly derived from biological gene duplication processes. In my own PushGP

system, architecture can evolve without such mechanisms; modules are created and executed by means of code-manipulation instructions that are always available, and the action of ordinary genetic operators (like crossover and mutation) can, by changing sequences of such instructions, change the architecture of the overall program (Spector and Robinson, 2002).

4.2 Evolved Developmental Processes

"Development" occurs at several levels in biology. As mentioned above, genetic mechanisms (in which we should include not only DNA but also RNA, enzymes, and proteins involved in gene expression, at least) themselves undergo complex forms of environmentally-mediated development in the normal course of cell activity. In addition, complex developmental processes at a larger (morphological) level of aggregation mediate the long transition from embryo to adult in multicellular organisms. Finally, behavioral/psychological developmental processes are often critical in producing adults capable of completing the life cycle of complex life forms.

Many of these developmental processes have already been incorporated, at some level of abstraction, into genetic programming systems. I have already mentioned several of the systems that use some form of "gene expression" development. In my own PushGP system programs are routinely self-manipulating, and the sequence of primitive instructions that is eventually executed can be very difficult to predict from the surface structure of the genome (program) prior to "development."

Morphological development has also appeared in various forms, usually when the execution of an evolved program builds a secondary structure which is then responsible for the desired problem-solving behavior. This is analogous to the construction, by a biological genome, of a body which is then responsible for behavior in the world (although this is really a flawed analogy, as indicated by the quotes from Keller above). Examples include Koza's work on the evolution of programs that build electrical circuits (Koza et al., 1999) and my own work on the evolution of programs that build quantum gate arrays (Barnum, Bernstein, and Spector, 2000).

Behavioral or psychological development has been incorporated in many systems that evolve structures that subsequently undergo "learning" processes as they interact with their problem environments. The most salient examples of this approach are systems that build neural networks that are then trained on problems from their target domains using

standard neural network learning algorithms; note that many of these systems incorporate both morphological and behavioral development.

4.3 Evolved Diversification

As indicated in the quotes from Keller above, in biology the mechanisms of diversification are themselves under genetic control and it appears clear that the mechanisms of evolution have thereby themselves evolved.

Within genetic and evolutionary computation there is a long tradition of systems in which mutation rates are genetically encoded. Within genetic programming more specifically there have been several explorations of "Meta-GP," in which not only the rates but also the algorithms for diversification are genetically encoded and can therefore evolve (Schmidhuber, 1987; Edmonds, 2001). In these systems co-evolving populations of program-manipulation programs are used in lieu of traditional genetic operators to produce the offspring of the individuals in the primary (problem-solving) population.

In my own recent work on "autoconstructive evolution" with the Push programming language, programs construct their own offspring using code-manipulation instructions that are available in the instruction set along with problem-related primitives. In these system the "genetic operators" are just the parts of individuals (possibly interleaved with other parts of individuals) that produce offspring — when a program is run it may do something that confers fitness (for example, providing correct answers to symbolic regression fitness cases, or navigating an agent toward food, etc.) *and* it may also produce a child. The code for the child may be produced in any computable way by the parent, possibly using the code of the parent, the code of other individuals, randomly generated code, etc. The Pushpop system grafted these concepts onto a fairly traditional genetic programming algorithm (Spector and Robinson, 2002; Spector, 2002). In more recent work my colleagues and I have built autoconstructive evolution systems embedded in 3D virtual worlds (in Jon Klein's BREVE system (Klein, 2002), the latest development versions of which include a Push language interpreter; see (Spector and Klein, 2003)).

4.4 Geography

Geography plays a critical role in evolutionary biological theory. Most theories of speciation rely on notions of geographic isolation, location-based ecological niches, and clinal variation of species characteristics across gradients of temperature and other environmental features. In-

deed, the interaction between geography and evolutionary processes has been an area of exploding interest in the last decade, referred to by some as "phylogeography" (Avise, 2000).

These concepts had little direct influence on early work in genetic and evolutionary computation, although the concept of "demes" (local breeding populations, between which migration occasionally occurs) provides a coarse-grain analog that has had considerable impact (see e.g. Fernández et al., 2003). More recently, however, concepts from artificial life models (in which spatial distribution is often modeled explicitly) have influenced work in genetic programming. One example is work by Punch and Rand combining genetic programming with elements from Holland's Echo architecture and Brooks's subsumption architecture (Punch and Rand, 2000). Another example is my work on evolution within spatially continuous 3D virtual worlds (Spector and Klein, 2003). In such environments, many other biological concepts that had previously played little or no role in genetic and evolutionary computation also come into play. For example, in many ALife-derived systems different individuals have different life expectancies, and their reproductive behaviors may change based on age and environmental circumstances.

One might expect, based on the importance that concepts of spatial distribution and geography have had in biological theory, that the influence of these concepts on genetic programming practice will continue to increase.

5. To Theory!

What kinds of mathematical theory would help us to analyze, understand, and predict the behavior of genetic programming systems that exhibit the challenges outlined above? There is a sense in which current theory already "handles" all of these challenges; for example even with all of the complications raised above, a genetic programming system can still be viewed as a markov chain (Poli, Rowe, and McPhee, 2001) and related theoretical apparatus can still be applied. But this is an unsatisfactory answer for at least two reasons. First, while it is conceivable that existing theoretical frameworks could be extended to meet these challenges this would require considerable effort and time (Poli, personal communication). Second, I suspect that the resulting theories, if generalized to handle such radical extensions to the basic technique, will be so general that they will cease to have explanatory power.

Are there alternatives? I believe that there are. The obvious place to look for mathematical theory that might be applicable to a field heading toward biology is biology itself. While relations between evolutionary

biology and mathematics have not always been harmonious (see for example Keller, 2002), there have nonetheless emerged many mathematical theories of aspects of biological evolution. These have often provided valuable insights into the nature and dynamics of evolving populations even when they ignored seemingly important aspects of biological systems (such as development, which was ignored in early population biology) and even when they were based on naive models of the molecular foundations of genetics (such as the assumption that individual traits are determined by individual genes).

Biologists have invented mathematical or quasi-mathematical theories of diversification, development, and the relations between genetics and geography. The current explosion of work in genomics is producing a wealth of mathematical theory concerning DNA "code structure" and gene expression (see for example Karp, 2002). There seems, therefore, to be a great deal of theory available for transplantation into genetic programming, suitable for addressing the specific challenges outlined above.

What differentiates mathematical theory in biology from the existing mathematical theory in genetic programming? I believe that the primary difference is that biologists (even mathematical biologists) begin with large scale phenomena (populations of complex organisms) and work downward to the mechanisms out of which these phenomena are built, while genetic programming theorists begin with a set of low level mechanisms and work upward to the large scale phenomena exhibited by our systems.

In genetic programming we have the luxury of *knowing* the exact nature of the mechanisms since we construct them, while in biology many of the mechanisms are still unknown. But this luxury is fleeting and perhaps illusive as well. As discussed above, the mechanisms used in genetic programming are now in considerable flux. In addition, while we may have complete knowledge of the mechanisms at the lowest level of abstraction this may not be the level at which the most fruitful theories can be built. For example, if a system incorporates developmental processes and adaptive representations then it is possible that mechanisms at a somewhat higher level, for example "body plans" and "regulator genes" will be essential to understanding its behavior. It is not obvious that such mechanisms will emerge from bottom-up theories based on, for example, a schema analysis of programs containing code-manipulation instructions.

Biology, for its part, has always had the luxury of truly interesting large scale phenomena to study, and biologists have built mathematical frameworks capable of providing useful generalizations of complex adap-

tive systems even in the face of ignorance about the ways in which those systems are implemented. This is, by itself, a good reason for genetic programming theorists to look to biological theory for inspiration. An additional reason is the fact that many of the new features being added to genetic programming systems originate in biology.

One example of the theoretical move advocated here, already underway in the community, is the empirical study of diversity and diversification in genetic programming systems. The study of diversity has deep roots in biology, which ought to provide context for these studies. In addition, many of the relevant questions can be asked in a way that transcends implementation details. Measures of diversity have been adopted by several researchers (a nice survey is available in Burke, Gustafson, and Kendall, 2002). In addition, researchers have begun to explore relations between diversification and adaptation; one of my studies, for example, showed that adaptive populations of endogenously diversifying Pushpop programs are reliably diverse (Spector, 2002). But this is only a start.

6. Prospects

It is possible that biological tools will not actually help us to understand our genetic programming systems. One possibility is that mathematical models that are sufficiently rich to capture the most important features of complex adaptive evolutionary systems, whether organic or technological, will be too complex or abstract to help *humans* understand how and why the systems behave as they do. This is a depressing thought, but I have little more to say about it one way or the other.

Another possibility is that the "right" theory for genetic programming (however extended) will bear little or no similarity to current theories in biology. One reader of a draft of this essay agreed with the critique of current genetic programming theory but felt that biology had unfortunately little to offer; for example, she found in her own work that most of the mathematical apparatus of population genetics, while useful in analyzing simple systems, failed to scale up to the complexity found in applied genetic programming work. As noted earlier, however, biology is in the midst of major transitions and there are bound to be substantial theoretical innovations in the coming years. My contention is simply that wherever biology goes, genetic programming should follow.

Yet another possibility is that the right theory for genetic programming will bear no resemblance to current *or future* theories in biology. After all, the critics will say, genetic programming is a problem solving technology, while biological evolution is not; the similarities are su-

perficial and should remain so. If so, then so much the worse for genetic programming theory; it will have to borrow its fundamentals from other fields (perhaps thermodynamics; Adami, 1997) or invent them from whole cloth.

In any event, it is probably worth noting that if the goal is *human understanding* of genetic programming then there may also be alternatives to mathematics. One alternative is understanding via visualization; often the human visual system can discover high-level, emergent properties for which we have, as yet, no formal theory. Genetic programming researchers have always used the traditional forms of scientific visualization (fitness graphs, statistical plots, etc.), but if the conjecture of this chapter is correct, and genetic programming is indeed moving toward biology, then other forms of visualization suggest themselves. In particular, insofar as future systems, like biological systems, exist and function in 3D (virtual) spaces, direct observation of 3D geometry as it evolves over time — a sort of virtual microscope for virtual ecosystems — may provide essential insights. This approach has already proven valuable in the field of artificial life (e.g. Spector and Klein, 2002) and it might be expected to have more relevance to genetic programming the closer genetic programming moves to biology.

Acknowledgments

Apologies to John Locke for the title. Riccardo Poli provided inspirational comments, though he bears no responsibility for the uses to which the inspiration has been put. Bill Langdon, Una-May O'Reilly, and Anjun Zhou provided thoughtful reviews that raised more good questions and issues than I could address here. This effort was supported by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30502-00-2-0611, and by NSF grant EIA-0216344. Support was also provided by Hampshire College for the Institute for Computational Intelligence.

Notes

1. http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html

2. The term "genetics" was introduced by William Bateson in 1906, and the term "gene" was introduced three years later by Wilhelm Johannsen (Keller, 2000).

References

Adami, C. (1997). *An Introduction to Artificial Life*. Telos Press.

Altenberg, L. (1994). "The Evolution of Evolvability in Genetic Programming." In Kinnear, K.E. Jr., ed. *Advances in Genetic Programming*. The MIT Press. pp. 47–74.

Avise, J.C. (2000). *Phylogeography: The History and Formation of Species*. Harvard University Press.

Bäck, T., and Schwefel, H.-P. (1995). "Evolution Strategies I: Variants and their computational implementation." In *Genetic Algorithms in Engineering and Computer Science*. John Wiley & Sons Ltd.

Banzhaf, W. (2003). Artificial Regulatory Networks and Genetic Programming. In *Proceedings of the Workshop on Genetic Programming, Theory and Practice*. Kluwer.

Barnum, H., Bernstein, H.J., and Spector, L. (2000). "Quantum circuits for OR and AND of ORs ." *Journal of Physics A: Mathematical and General*, 33(45), pp. 8047–8057.

Burke, E., Gustafson, S., and Kendall, G. (2002). "A Survey And Analysis Of Diversity Measures In Genetic Programming." In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, CA: Morgan Kaufmann. pp. 716–723.

Downing, K.L. (2001). "Reinforced Genetic Programming." *Genetic Programming and Evolvable Machines*, 2(3), pp. 259–288.

Edmonds, B. (2001). "Meta-Genetic Programming: Co-evolving the Operators of Variation." *Elektrik, the Turkish Journal of Electrical Engineering and Computer Sciences*, 9(1), pp. 13–29.

Fernández, F., Tomassini, M., and Vanneschi, L. (2003). "An Empirical Study of Multipopulation Genetic Programming." *Genetic Programming and Evolvable Machines* 4(1), pp. 21–51.

Ferreira, C. (2001). "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems." *Complex Systems*, 13(2).

Hansen, J.V. (2003). "Genetic Programming Experiments with Standard and Homologous Crossover Methods." *Genetic Programming and Evolvable Machines* 4(1), pp. 53–66.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems.* The MIT Press.

Karp, R.M. (2002). "Mathematical Challenges from Genomics and Molecular Biology." *Notices of the AMS*, 49(5), pp. 544–553.

Keijzer, M. (1996). "Efficiently Representing Populations in Genetic Programming." In Angeline, J., and Kinnear, K.E. Jr., eds., *Advances in Genetic Programming 2*. The MIT Press. pp. 259–278.

Keller, E.F. (2000). *The Century of the Gene*. Harvard University Press.

Keller, E.F. (2002). *Making Sense of Life: Explaining Biological Development with Models, Metaphors, and Machines.* Harvard University Press.

Keller, R.E., and Banzhaf, W. (2001). "Evolution of Genetic Code on a Hard Problem." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann. pp. 50–56.

Kinnear, K.E. Jr. (1994). "Alternatives in Automatic Function Definition: A Comparison of Performance." In Kinnear, K.E. Jr., ed. *Advances in Genetic Programming*. The MIT Press. pp. 119–141.

Klein, J. (2002). "BREVE: a 3D Environment for the Simulation of Decentralized Systems and Artificial Life." In Proceedings of Artificial Life VIII, The 8th International Conference on the Simulation and Synthesis of Living Systems. The MIT Press. pp. 329–334.

Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press.

Koza, J.R. (1995). "Gene Duplication to Enable Genetic Programming to Concurrently Evolve Both the Architecture and Work-Performing Steps of a Computer Program." In *IJCAI-95 Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann. pp. 734–740.

Koza, J.R., Andre, D., Bennett, F.H. III, and Keane, M. (1999). *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman.

Margulis, L. (2000). *Symbiotic Planet*. Basic Books.

Maynard Smith, J., and and Szathmáry, E. (1999). *The origins of life*. Oxford University Press.

Nikolaev, N.I., Iba, H., Slavov, V. (1999). "Inductive Genetic Programming with Immune Network Dynamics." In Spector, L., et al., eds. *Advances in Genetic Programming 3*. The MIT Press. pp. 355–376.

Nordin, P., Banzhaf, W., Francone, F.D. (1999). "Efficient Evolution of Machine Code for CISC Architectures using Instruction Blocks and Homologous Crossover." In Spector, L., et al., eds. *Advances in Genetic Programming 3*. The MIT Press. pp. 275–299.

Poli, R. (2001). "General Schema Theory for Genetic Programming with Subtree-Swapping Crossover." In J.F. Miller et al., eds. *Genetic Programming, Proceedings of EuroGP'2001*. Springer-Verlag. pp. 143–159.

Poli, R., Rowe, J.E., and McPhee, N.F. (2001). "Markov Chain Models for GP and Variable-length GAs with Homologous Crossover." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. San Francisco, CA: Morgan Kaufmann. pp. 112–119.

Poli, R., and Langdon, W.B. (1999). "Sub-machine-code Genetic Programming." In Spector, L., et al., eds. *Advances in Genetic Programming 3*. The MIT Press. pp. 301–323.

Punch, W.F., and Rand, W.M. (2000). "GP+Echo+Subsumption = Improved Problem Solving," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000).* San Francisco, CA: Morgan Kaufmann. pp. 411–418.

Schmidhuber, J. (1987). "Evolutionary principles in self-referential learning." Diploma thesis, Institut für Informatik, Technische Universität München.

Spector, L. (1996). "Simultaneous Evolution of Programs and their Control Structures." In Angeline, P.J., and Kinnear, K.E. Jr., eds., *Advances in Genetic Programming 2.* The MIT Press. pp. 137–154.

Spector, L. (2002). "Adaptive populations of endogenously diversifying Pushpop organisms are reliably diverse." In *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems.* The MIT Press. pp. 142-145.

Spector, L., and Robinson, A. (2002). "Genetic Programming and Autoconstructive Evolution with the Push Programming Language." *Genetic Programming and Evolvable Machines*, 3(1), pp. 7–40.

Spector, L., and Klein, J. (2002). "Evolutionary Dynamics Discovered via Visualization in the BREVE Simulation Environment." In *Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems.* Sydney, Australia: University of New South Wales. pp. 163–170.

Spector, L., and Klein, J. (2003). "Emergence of Collective Behavior in Evolving Populations of Flying Agents." To appear in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003).*

Stoffel, K., and Spector, L. (1996). "High-Performance, Parallel, Stack-Based Genetic Programming." In *Genetic Programming 1996: Proceedings of the First Annual Conference* The MIT Press. pp. 224–229.

Teller, A. (1999). "The Internal Reinforcement of Evolving Algorithms." In Spector, L., et al., eds. *Advances in Genetic Programming 3.* The MIT Press. pp. 325–354.

Wu, A.S., and Garibay, I. (2002). "The Proportional Genetic Algorithm: Gene Expression in a Genetic Algorithm." *Genetic Programming and Evolvable Machines* 3(2), pp. 157–192.