Automatic Programming with Grammatical Evolution

Michael O' Neill
Dept. of Computer Science And Information Systems
University of Limerick
Ireland
Michael.ONeill@ul.ie

ABSTRACT

The aim of this work is to develop an Automatic Programming system drawing inspiration from biological genetic systems. Grammatical Evolution (GE), a system which can generate code in any language from variable length binary strings is the result of this investigation. Using a language definition in the form of a Backus Naur Form grammar permits the generation of programs in any language, of arbitrary complexity, through a genotype to phenotype mapping process.

Grammatical Evolution (GE) is a grammar based, variable length, linear genome system which is capable of generating code in any language. Rather than the functions and terminals associated with GP [5], GE takes a BNF specification of a language, or subset thereof, from which it can subsequently generate code. The BNF is used in a genotype to phenotype mapping process which produces a program from the genotypic binary string. Various genetic programming systems using grammars, and genotype to phenotype mapping have been developed prior to GE by other researchers, for examples see [3] [1] [15] [16] [2] [11], however GE's unique mapping process is it's distinguishing feature, see [10] for a description.

Although, at present the search technique used by the system is a Genetic Algorithm, conceivably any technique which can search variable length binary strings could be employed. Investigations in other possible search methods, and a comparison of their performances, has yet to be conducted.

In order to facilitate the evolutionary process in this system, a linear degenerate genetic code, along with a genotype to phenotype mapping has been used. The degenerate code means that there is a complex many to one mapping, i.e. codons of different values can represent the same production rule of the BNF definition. It is envisaged that these two principles serve to maintain validity of the phenotypic programs whilst allowing an unconstrained search to be performed upon the genotypic binary string. According to the neutral theory of molecular evolution [4] these principles should also serve to maintain genetic diversity within the population. This is due to the occurrence of *neutral mutations*, mutations which have no effect on the fitness of the phenotype. Following on from Kimura's theory the idea of *neutral networks* has been developed. A neutral network is comprised of individuals in the search space which are separated by single-point neutral mutations. It has been shown that within such a fitness landscape with increasing degrees of neutrality the maximum fitness attainable in a population increases [6]. The implications for GE have yet to be empirically investigated, however the suggestion is that by increasing the degeneracy in the code, the maximum obtainable fitness should rise.

During the mapping process it is possible to generate invalid (incompletely mapped) individuals, and so to counteract the transmission of these individuals to subsequent generations a Steady State replacement mechanism is employed [14]. A consequence of the Steady State method is its tendency to maintain fit individuals at the expense of less fit, and in particular, invalid individuals as these are given the lowest possible fitness value.

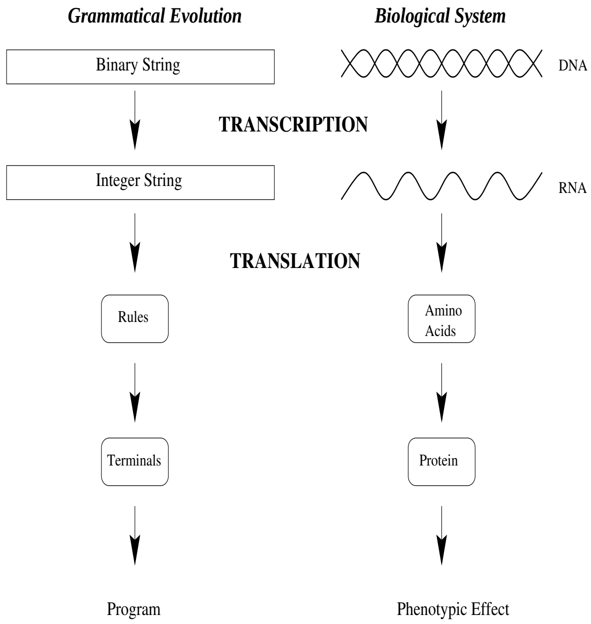

Figure 1 A comparison between Grammatical Evolution and a Biological System.

GE has been successfully applied to a number of diverse problem domains such as, Symbolic Regression [12], Finding Trigonometric Identities [13], the Santa Fe Trail [10], and evolving Caching Algorithms [9]. The results compared favorably with systems such as GP, and in the case of the Santa Fe Trail problem have been shown to outperform GP [10]. An in depth analysis into the system is currently being undertaken, and some initial results can be seen in [8], and [7].

References

[1] Gruau, F. 1994. *Neural Network synthesis using cellular encoding and the genetic algorithm.* PhD Thesis from Centre d'etude nucleaire de Grenoble, France.

[2] Horner, H *A C++ class library for GP*. Vienna University of Economics.

[3] Keller, R. & Banzhaf, W. 1996. Genetic Programming using Genotype-Phenotype Mapping from Linear Genomes into Linear Phenotypes. In *Proceedings of Genetic Programming 1996*, pages 116-122. MIT Press.

[4] Kimura, M. 1983. The Neutral Theory of Molecular Evolution. Cambridge University Press.

[5] Koza, J. 1992. *Genetic Programming*. MIT Press.

[6] Newman M.E.J., Engelhardt Robin. 1998. Effects of neutral selection on the evolution of molecular species. In *Proc. R. Soc. London B 265*, pages 1333-1338.

[7] O'Neill M., Ryan C. 1999. Genetic Code Degeneracy: Implications for Grammatical Evolution & Beyond. In *Proceedings of the Fifth European Conference on Artificial Life 1999*. To Appear.

[8] O'Neill M., Ryan C. 1999. Under the Hood of Grammatical Evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference 1999.*

[9] O'Neill M., Ryan C. 1999. Automatic Generation of Caching Algorithms. In *Proceedings of EUROGEN 99.*

[10] O'Neill M., Ryan C. 1999. Evolving Multi-line Compilable C Programs. In *Proceedings of the Second European Workshop on Genetic Programming 1999*.

[11] Paterson N., Livesey M. 1997. Evolving caching algorithms in C by GP. In *Proceedings of Genetic Programming 1997*, pages 262-267. MIT Press.

[12] Ryan C., Collins J.J., O'Neill M. 1998. Grammatical Evolution: Evolving Programs for an Arbitrary Language. In *Proceedings of the First European Workshop on Genetic Programming*, pages 83-95. Springer-Verlag.

[13] Ryan C., O'Neill M., Collins J.J. 1998. Grammatical Evolution: Solving Trigonometric Identities. In *Proceedings of Mendel '98*, pages 111-119.

[14] Ryan C., O'Neill M. 1998. Grammatical Evolution: A Steady State Approach. In *Late Breaking Papers, Genetic Programming 1998*, pages 180-185.

[15] Whigham, P. 1996. Search Bias, Language Bias and Genetic Programming. In *Genetic Programming 1996*, pages 230-237. MIT Press.

[16] Wong, M. and Leung, K. 1995. Applying logic grammars to induce subfunctions in genetic programming. In *Proceedings of the 1995 IEEE conference on Evolutionary Computation*, pages 737-740. USA:IEEE Press.