Attributed Grammatical Evolution using Shared Memory Spaces and Dynamically Typed Semantic Function Specification

James Vincent Patten and Conor Ryan
Biocomputing and Developmental Systems Group,
Department of Computer Science and Information Systems,
University of Limerick
{james.patten,conor.ryan}@ul.ie

Keywords: Computer Science and Information Technology

Abstract

Grammatical Evolution (GE) is an evolutionary computation (EC) technique which can automatically generate programs in any language. GE systems use Context Free Grammars (CFG) which have expressive limitations and our research is into ways to reduce these limitations and improve the performance of GE.

1. Introduction

Grammatical Evolution (GE) [3] has been successfully applied to solve a wide range of problems across a diverse set of domains. GE operates by producing populations of potential solutions, to a predefined problem, using combinations of symbols specified by a number of rules in Backus-Naur Form (BNF), a convenient way of describing a Context Free Grammar (CFG). The fitness or potential of each individual to address the predefined problem is assessed and this drives the evolutionary cycle of GE.

The symbol sequences of CFG rules provide a means of encoding program syntax but a CFG cannot encode program semantics. Access to semantic information could help guide the generation of more meaningful and precise programs.

2. Limitations of cfg

As a direct result of CFGs inability to encode semantics GE systems may introduce individuals into the population which are semantically invalid. A commonly used approach for handling invalids is to add a test during fitness evaluation which assigns the worst possible fitness to any invalids detected. This approach relies on the fact that a poor fitness will greatly reduce the probability of an invalid being selected for reproduction during evolution, meaning that its genetic material will fail to be used in the creation of the new populations and therefore "die off".

While this is a widely used approach for dealing with invalids research has shown that allowing invalids into a population does have negative consequences for GE [1], therefore an alternative approach is required. One approach is to prevent the generation of invalids prior to their introduction to the population. To do this GE needs to support a grammar which can encode semantics. Knuth [4] proposed such a grammar which uses attributes (inherited and synthesized) and functions to encode semantics. This grammar is commonly known as an Attribute Grammar (AG).

3. Attribute grammar GE system (AGES)

CFG rules contain two types of symbols, non-terminals and terminals. During a process known as mapping GE uses rules to produce tree type structures (derivation trees) with nodes containing either non-terminals or terminals. Reading only terminal nodes of a derivation tree gives a "parse tree" and this is the program used during the fitness evaluation process.

When used with GE an AG adds context to grammar symbols by passing information between derivation tree nodes during mapping, down using inherited attributes and up or across using synthesized. Semantic functions use this information to prevent the generation of semantically invalid individuals.

While we are not the first to suggest an AG supporting GE system we do feel that our new AG-GE system (AGES) has the greatest chance of maintaining the existing levels of accessibility found in traditional CFG GE systems.

Our AGES core is implemented in C++ and has derivation tree structures designed to support addition of attributes, in shared memory locations, allowing for easy information passing between tree nodes. AGES supports semantic function specification using Python and includes a C++/Python interface which allows semantic functions create/update attributes when the grammar rules are being applied. We have carried out a number of experiments using AGES and the initial results have been positive [4].

4. References

[1] Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Genetic programming, pp. 70-82. Springer (2003)
[2] Knuth, D.E.: Semantics of context-free languages. Mathematical systems theory 2(2), 127-145 (1968)
[3] O'Neil, M., Ryan, C.: Grammatical evolution. In: Grammatical Evolution, pp. 33 - 47. Springer (2003)
[4] Patten, J.V., Ryan, C.: Attributed Grammatical Evolution using Shared Memory Spaces and Dynamically Typed Semantic Function Specification. In: Proceedings of the 18th European Conference on Genetic Programming. Forthcoming