Grammar Evolution as Epistemic Computation

Rüya Sanver
BcS, Computer Engineering
Muğla Sıtkı Koçman University
ruyasanver@posta.mu.edu.tr

Abstract: Grammatical Evolution is an evolution inspired algorithm for generating symbolic expressions guided by formal grammar. While it excels in tasks like symbolic regression, the expressive power of Grammatical Evolution is constrained by the use of formal grammars. This paper argues that these limits are not just technical, but epistemological as well. It highlights how formal systems, like those in mathematics and computation, can fail to represent certain truths. Inspired by evolutionary epistemology, we explore recent advancements in Grammatical Evolution that allow the language itself to evolve and adapt, including grammar mutations, probabilistic grammar models, and symbiogenesis. These techniques allow modification of not only solutions but also the structure of representations. We frame this shift from optimization to exploration, where Grammatical Evolution becomes capable of adapting to unknown domains by evolving the language of discovery.

Keywords: Grammatical Evolution, Symbolic Regression, Meta-Grammars, Grammar Mutation, Computational Epistemology, Evolutionary Computation.

---

1) Introduction: From Determinism to Evolving Structure

In traditional symbolic systems, knowledge is discovered within strict formal frameworks. From automata to programming languages, deterministic methods shaped how we encode thought, find meaning and explore computational possibilities. Grammatical Evolution (GE), introduced by O'Neil and Ryan (Neill & Ryan, 2001), changed this rigidity by introducing evolutionary principles into the synthesis of programs. GE generates solutions using genotype to phenotype mapping guided by formal grammars, oftentimes defined using Backus-Naur Form (BNF) (JW). Genotypes are, in practice, binary strings that are mapped to phenotypes which are programs or expressions This formation allows syntactically valid structures in symbolic regression tasks. The goal of symbolic regression is to discover explainable mathematical models from data.

However, from an epistemological stance, the use of static formal grammars imposes fundamental limits (Greibach, 1981). Formal grammars define not only the valid syntactic structure but also what is expressible using them, therefore what can be discovered. As an example, if a "true" solution which might be the global best, lies outside the representational limits of a grammar, GE will fail to explore it. This limit mirrors Gödel's famous Incompleteness Theorems, which shows even consistent, well-defined formal systems contain truths they fail to express (Godel, 1962). In symbolic regression, this means a correct model might be unreachable not because of the algorithm, but because of the grammar itself.

This paper focuses on exploring recent advancements in GE that tries to overcome this constraint. Through mechanisms like grammar mutation, probabilistic grammar evolution and meta-grammatical systems, the language of evolution, which is the grammar, itself can evolve. In light of this, GE begins to resemble an epistemic engine, it becomes not just a method for

regression, but a computational metaphor for the evolution of structures.

2) Background: GE and Symbolic Regression

Grammatical Evolution is an algorithm that generates solutions by evolving numerical genomes. These genomes are mapped to symbolic expressions using formal grammar, mostly defined in BNF. This mapping also known as genotype-to-phenotype mapping, ensures that the generated outputs are always syntactically valid. In practical terms, GE enables the automatic generation of code, formulas or mathematical models based on rules defined by grammar.

For a long time, formal grammars have been under inspection of linguists and computer scientists who work on computational theory. We owe a lot of our understanding to Noam Chomsky who was the first person to formalize generative grammars in 1956 (Chomsky, 1956). In his study, Chomsky classified generative grammars into types based on their expressive power, known as Grammar Hierarchy. His work in this intersection field, helped clarify our understanding of languages.

Formal grammars are structured and static rule sets used to define the syntax of programming languages and symbolic expressions (Greibach, 1981). Because they strictly constrain which forms are valid, they are well-suited for guiding evolutionary algorithms.

For example, in symbolic regression, a grammar might define valid mathematical functions or operations. This guarantees that any expression evolved, follows correct mathematical syntax.

Figure 1 shows a BNF grammar defining a minimal language for arithmetic expressions.

$$< expr >::= < expr >< op >< expr > \mid < var > \mid < const >$$

$$< op > ::= + \mid - \mid * \mid /$$

$$< var > ::= x \mid y$$

$$< const >::= 1 \mid 2 \mid 3$$

Figure 1 - A simple BNF grammar generating basic arithmetic expressions.

Formal grammars constrain and control the search space, ensure syntactical validity of results and in some cases arguably has the potential to result in semantically better results.

Symbolic Regression proposed by John Koza (Koza, 1994), is a modelling process of discovering a mathematical expression that best fits a set of input-output data. Unlike traditional regression problems, which usually focus on fitting coefficients into a predefined equation, symbolic regression searches for the equation itself. Hence can be interpreted as being a more complex task (Koza, 1990).

GE is famously well-suited for symbolic regression because the grammar can guide the form of the expressions being evolved, while also ensuring they are interpretable and syntactically coherent (Kronberger et al., 2024).

In this paper, we propose that the usage of static grammars is a major limitation of GE. A grammar defines what is possible to express. A given input string must follow specific rules. Everything is syntactically valid but also structurally constrained, both a blessing and a curse. If the grammar is too structurally rigid, you will never discover certain solutions even if they fit better. Therefore, if the optimal solution is not contained inside the BNF grammar, symbolic regression will never evolve into it.

But what if the grammar is too limited or the problem needs new kinds of expressions. If a solution lies outside of

what the grammar allows, GE will fail to explore it (Orzechowski et al., 2018).

This means the creativity of the algorithm is significantly constrained by its language. In complex or unknown domains, which are usually the reason we opt for genetic algorithms, this can severely limit the expressivity of symbolic regression.

However, if we can evolve the grammar itself, we can introduce a statistical, stochastic layer to how structure evolves.

This idea of evolving grammar was first seen in O'Neil and Ryan's work (O'Neill & Ryan, 2004). Meta-grammatical GE evolves the shape of the search space itself as opposed to the traditional GE which evolved within the search space defined by the grammar (O'Neil & Ryan, 2003).

This epistemological switch transforms the search task from deterministic to an adaptive exploration.

3) Beyond Static Grammars: Meta-Grammars, Grammar Mutation, and Symbiogenesis

Meta-Grammatical Genetic Evolution refers to the systems that evolve not just the content of expressions but, the very grammar that defines the search space (Carvalho et al., 2023). In contrast to traditional GE, which operates within a fixed, static structure, meta-grammars allow the structure to adapt over generations. This approach enables not only new solutions but completely new forms of reasoning to emerge within the GE tasks.

A metaphorical similarity can be made with a player not solely strategizing within fixed rules but reshaping the rules themselves to allow new forms of play. In our context, rigid grammars reflect the constrained formal systems of classical computation, and GE's evolving grammars reflect the need for models to adapt as understanding deepens.

Grammar Mutation is one of the simplest ways to evolve a grammar (Ryan et al., 2018). This method involves stochastic modifications to produce new rules. Modifications can either be done at runtime or between generations. But unlike a random addition, this modification can be done in a context-aware, purposeful way.

The involvement of Grammar Mutation in GEs, introduces a statistical or probabilistic layer into the structure of search, allowing algorithm to escape local optima created by rigid formal grammars (Mégane et al., 2022).

Another approach which can be used in the evolution of grammars is Symbiogenesis (Gontier, 2006). Inspired once again by biology, Symbiogenesis refers to the synthesis of separate organisms to form a new, combinational whole. In context of GE, this approach can be translated as fusing multiple grammars into a composite rule structure. Instead of evolving one grammar, we evolve an ecosystem of grammars, which can interact, merge or compete with one another. This method creates a combinational way of increasing expressivity.

On an epistemological note, developing new languages to describe and model the world better echoes philosophical ideas from Kuhn to Gödel. In GE, grammar evolution becomes a computational analogy for evolving the representations that make express truths.

These emerging approaches, grammar mutation, meta-grammars, symbiogenesis, are more than an algorithmic trick. They highlight a shift in how we conceptualize problem solving, as a continuous redefinition of representation and not traversal of fixed structures. Just like how natural evolution shapes and reshapes organisms and environments continuously, meta-grammatical evolution reshapes both solutions, and the frameworks that describe the solution space. This adaptability brings GE closer to modelling the reasoning itself,

not just as computation but as an evolving epistemic process.

4) Formalism, Determinism, and the Evolution of Computational Language

Formal grammars like BNF, by design, encode structure through deterministic syntax. In computer science a simple theoretical machine used to recognize regular languages is called Deterministic Finite Automaton (DFA) (Rabin & Scott, 1959). DFA works by interpreting input symbols and changes states according to the fixed rules. Determinism comes from the fact that at any time, given a certain input, DFA has only one possible next state.

A connection of similarity can be achieved considering DFAs and formal grammars both define what sequences are syntactically valid based on fixed rules. When GE uses a static grammar, we can only generate what grammar allows. Therefore, it is not surprising to see formal systems' significant role in programming, automata, and GE.

The disadvantage of fixed structure comes to the surface when we are dealing with a complex domain. In systems we can only partly interpret the rules, or even fully fail to understand, the modelling problems become abstract and uncertain. (Kronberger et al., 2024; La Cava et al., 2021; Orzechowski et al., 2018) In the context of symbolic regression for example, if the grammar fails to describe a certain kind of solution, then it will not be explored, how optimal it might be.

In early 1900s, David Hilbert pioneered a belief, a program. The program known as the Hilbert Program (Zach, 2007) had a mission to build a complete, consistent, formal system that can prove all mathematical truths.

This view affected and was deeply rooted in what is known as formalism in philosophy of mathematics. Formalism is the view that holds mathematics, and logic can be considered a consequence of alphanumeric sequences of symbols.

A reformative paradigm shift occurred when in 1931, Kurt Gödel proved the famous Incompleteness Theorems (Godel, 1962) [i] . On a very concentrated summary, what Incompleteness said was that any complete and sufficient formal system is incomplete. Incompleteness in this context refers to the lack of the systems proving their own consistency.

This ambitious study was revolutionary for computation, algorithms, logic and even grammars (Batzoglou, 2024). In parallel to this paradigm shift, we can conclude that formal grammars are subject to similar expressive incompleteness. Just as formal arithmetic cannot capture all mathematical truths, static grammars in GE are limited in expressivity (Adachi, 1990). For example, symbolic regression bounded by static grammars cannot reach all valid solutions.

The power of discovery of a symbolic regression then can be linked to two core needs: the need for the data used to be efficient and the need for expressive power of the grammar. As a conclusion, even perfect data cannot be modelled if grammar cannot express all solutions.

No computer science study would be complete without the mention of Turing Machines. Inspection of Turing Machines in GE context can provide a historical and metaphorical insight. Alan Turing in 1936, produced a theoretical model of computation that defines what problems are computable. A Turing Machine in practice, processes symbols one at a time, using finite set of rules and memory (defined as a tape in original paper) to solve problems (Turing, 1938). Turing Machines has been the baseline of computation ever since (Cooper, 2017). Soon after its publication, a limitation was discovered of such state-based computation, it failed to provide solutions to certain problems, like the famous

Halting Problem. Just three years after Alan Turing proposed Oracle Machines, hypothetical machines that can solve what Turing Machines failed to solve (Turing, 1939). In principle Oracle Machines would be able to solve certain problems by querying an external "oracle."

What sounds as a mystical oracle, as seen from all the science discussions on formalism to static systems and Hilbert to Incompleteness, was at its core a need for superior modelling understanding.

Meta-Grammatical GE is not an oracle in exact sense, but they are comparable. GE with evolving and adapting grammars can be linked to oracle-like external adaptation which step outside and revolve the fixed structure. While no system can transcend its own formal limits, an evolving system can explore changes on its form. Grammar evolution offers a computational response to the rigidity of formalism; instead of working within a fixed space of expressions, it allows the space to adapt to needs. Similar to how biological evolution reshapes both the phenotype and genetic instructions, evolving grammars in GE reshapes the symbolic structures that define the solution space.

This paradigm is more than philosophical and abstract. It has been followed by technical advancements in GE. Recent developments like meta-grammars, probabilistic grammars, stochastic grammar mutation and symbiogenetic fusion of grammars allow search spaces to expand, combine, and direct themselves to promising regions. These methods are practical solutions to the structural inflexibility by enabling symbolic regression to find models that would be unreachable under fixed syntax.

In this light, grammar evolution is not just an optimization tool but a mechanism which represents learning. Through these advancements GE moves closer to a reasoning model capable of adaptation.

5) Discussion & Conclusion

This paper has argued that GE with static grammars is fundamentally limited in its expressive power, and evolving grammars enable not only better solutions, but also richer models.

It is, however, important to note that while grammar evolution improves the performance of GE methods like symbolic regression, its deeper contribution lies in adapting the structure of its own language. By doing so, this approach models how systems evolve in real life, this mirrors scientific understanding itself.

This view is what is known as evolutionary epistemology, a field pioneered by Karl Popper and Nathalie Gontier (Gontier, 2006). It argued that knowledge grows through a process of variation, selection, and adaptation. In this light, evolving grammars are not just tools but symbolic analogues of science and meaning.

In this broader context, Grammatical Evolution becomes more than a technical method, it acts as an epistemic engine. A dynamic system that not only searches for solutions but adapts the language of solutions.

Through evolving both the structure and content, GE moves closer to computational exploration, not just optimizing within a known space. This approach promises a value in automated discovery, where machines could evolve new models or languages in response to data.

Looking forward, exciting directions can emerge for the field:

- Scientific modelling: Using grammar evolving GE to uncover interpretable models in science.
- Semantic grammars: Allowing GE to discover not just valid

expressions, but semantically rich representations.

- Hybrid systems: Combining GE with neural networks or differentiable programming to merge structure with statistical learning.

To conclude, evolving grammars point toward a kind of computational understanding, where machines do not just search for solutions but adapt the language in which those solutions are expressed. In a world where the problems we encounter are dynamic and our understanding incomplete, evolving the language of discovery may be one of the most important steps toward better systems.

Bibliography

Adachi, A. (1990). *Foundations of Computation Theory*. IOS Press, Incorporated.

Batzoglou, S. (2024). Gödel's Incompleteness Theorems. In *Introduction to Incompleteness: From Gödel's Theorems to Forcing and the Continuum Hypothesis* (pp. 35-54). Springer.

Carvalho, P., Mégane, J., Lourenço, N., & Machado, P. (2023). Context matters: adaptive mutation for grammars. European Conference on Genetic Programming (Part of EvoStar),

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113-124. https://doi.org/10.1109/TIT.1956.1056813

Cooper, S. B. (2017). *Computability theory*. Chapman and Hall/CRC.

Godel, K. (1962). On formally undecidable propositions of" Principia Mathematica" and related systems. In: Oliver & Boyd.

Gontier, N. (2006). Evolutionary epistemology and the origin and evolution of language: Taking symbiogenesis seriously. *Evolutionary Epistemology, Language and Culture: A Non-Adaptationist, Systems Theoretical Approach*, 195-226.

Greibach, S. A. (1981). Formal Languages: Origins and Directions. *3*(01), 14-41. https://doi.org/10.1109/mahc.1981.10006

JW, B. c The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference.

Koza, J. R. (1990). *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems* (Vol. 34). Stanford University, Department of Computer Science Stanford, CA.

Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, *4*, 87-112.

Kronberger, G., Burlacu, B., Kommenda, M., Winkler, S., & Affenzeller, M. (2024). *Symbolic Regression*. https://doi.org/10.1201/9781315166407

La Cava, W., Burlacu, B., Virgolin, M., Kommenda, M., Orzechowski, P., de França, F. O.,…Moore, J. H. (2021). Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems*, *2021*(DB1), 1.

Mégane, J., Lourenço, N., & Machado, P. (2022, 18-23 July 2022). Probabilistic Structured Grammatical Evolution. 2022 IEEE Congress on Evolutionary Computation (CEC),

Neill, M. O., & Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, *5*(4), 349-358. https://doi.org/10.1109/4235.942529

Orzechowski, P., Cava, W. L., & Moore, J. H. (2018). *Where are we now? a large benchmark study of recent symbolic regression methods* Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan.

https://doi.org/10.1145/3205455.3205539
O'Neil, M., & Ryan, C. (2003). Grammatical Evolution. In *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language* (pp. 33-47). Springer US. https://doi.org/10.1007/978-1-4615-0447-4_4
O'Neill, M., & Ryan, C. (2004). Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. European Conference on Genetic Programming,
Rabin, M. O., & Scott, D. (1959). Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, *3*(2), 114-125. https://doi.org/10.1147/rd.32.0114
Ryan, C., O'Neill, M., & Collins, J. (2018). *Handbook of grammatical evolution* (Vol. 1). Springer.
Turing, A. M. (1938). On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction. *Proceedings of the London Mathematical Society*, *s2-43*(1), 544-546. https://doi.org/https://doi.org/10.1112/plms/s2-43.6.544
Turing, A. M. (1939). Systems of logic based on ordinals. *Proceedings of the London Mathematical Society, Series 2*, *45*, 161-228.
Zach, R. (2007). Hilbert's program then and now. In *Philosophy of logic* (pp. 411-447). Elsevier.

[i] Original publication was in 1932 by Gödel. However the paper was re-translated to English and republished in 1039.