Generative Logic: Teaching Prolog as Generative AI in Art and Design

Christian Jendreiko[1]

[1]*HSD, University of Applied Sciences, Düsseldorf, Germany*

Abstract

This paper introduces a method for teaching Prolog, especially to BA and MA students in art and design faculties. The method is called Exploring Generative Logic (EGL).

The method is based on the concept of Generative Logic. Both methods are under development at the Faculty of Design, HSD, University of Applied Sciences, Düsseldorf. The basic idea of teaching Prolog with EGL is to present Prolog not as "another" programming language, but primarily as a logic AI. This perspective helps beginners gain a clear view of how Prolog works and focuses on understanding Prolog as a powerful generative AI, making it highly attractive in the domain of art and design. The author assumes that the EGL method can be easily adapted to other educational settings: education of non-programmers of all ages, including children in elementary or secondary schools, as well as students in other non-STEM curricula, to develop skills in logical thinking, digital literacy, AI literacy, and artistic literacy. The paper is divided into two parts: Part one provides an introduction to the basic ideas and objectives of the teaching method. Part two offers a description of the teaching method.

Keywords

Generative Logic, Teaching Prolog, Logic Programming, Generative AI, Generative Design, Symbolic AI, Digital Literacy, AI Literacy

1. Preliminaries

1.1. Basic idea

The basic idea behind the teaching method Exploring Generative Logic is to teach Prolog to non-programmers not as "another" programming language, but as a logic AI. It is a logic AI system that performs automated deduction and comes equipped with a formal language based on logic. The student can use this language to communicate with the AI, so that the logic AI derives output from what the student gives the system as input.

1.2. Target group

The target group for the EGL method consists of non-programmers, particularly BA and MA students in art and design. Many of them are heavy users of digital devices but have little to no experience in programming and are unfamiliar with concepts from computer science, mathematics, or logic.

1.3. Origin of the idea: The concept of Generative Logic

Generative Logic (GL) is a conceptual framework for the application of Prolog in the domain of art and design as a generative tool. It is being developed by the author at the HSD.

PEG 2024: 2^{nd} Workshop on Prolog Education, October, 2024, Dallas, USA.

christian.jendreiko@hs-duesseldorf.de (C. Jendreiko)
0009-0006-4302-6261 (C. Jendreiko)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-3799/paper9PEG2.0.pdf

Generative Logic is driven by 4 key ideas:

- GL puts its focus on the generative power of Prolog.
- GL interprets a Prolog System not as a language but as a logic AI that performs automated reasoning and comes equipped with a language that can be used by programmers and users to communicate with that AI in order to generate interesting aesthetic output.
- GL tries to foster the application of Prolog in the domain of art and design as a powerful generative tool.
- GL understands logic AI as a white box, human-friendly, and trustworthy alternative or complement to black box subsymbolic AI.

In its name, GL refers to the concept of Generative Aesthetics [7] a concept developed by the so-called Stuttgart school (Max Bense and his students Frieder Nake and Georg Nees) pioneering what is known today as generative design on a sound theoretical and critical basis. GL understands itself as a sub-class of this highly lively current design practice. It is a sub-class that is distinct from other methods within this field in making use of logic AI as the central generative tool. Aiming to show the generative power of automated logic inference, GL proves thereby the statement of Abraham A. Moles true, that every analytical machine can be used as a generative machine [6].

Exploring Generative Logic (EGL) is a teaching method grounded in Generative Logic. The goal is to empower students to explore the generative potential of Prolog within the realms of art and design. By delving into the generative capabilities of logic-based AI, students engage deeply with concepts of generative aesthetics while becoming familiar with logic programming, logic, mathematics, and computer science. They learn to use Prolog not only as a powerful generative tool but also as a means of fostering critical thinking.

The method of EGL is under development. The Winter Semester 2024/25 will be the second semester that I teach Prolog this way. Results of the seminar in the Summer Semester 2024 show that this approach is promising.

1.4. Why teaching Prolog not as a language but as an AI?

1.4.1. Reason No. 1: Making Prolog attractive for students

Introducing Prolog as an AI in the era of “generative AI” attracts students.

It makes them curious about what it is in comparison to currently popular subsymbolic AIs like LLMs.

Introducing Prolog as a generative device picks up the students right where they love to be the most: in a setting where they can create aesthetic objects.

Introducing Prolog as a classic “brain in a vat” species with which you can only communicate in an artificial language spurs the imagination of the students, making them feel part of a classic science fiction story about man-machine interaction. And because art & design students love Science Fiction, looking at Prolog as a “brain in a vat” makes working with Prolog fun and exciting.

1.4.2. Reason No. 2: Making Prolog easy to understand

Introducing Prolog as a logic AI instead of a language in the first place helps to avoid confusion for students in understanding what Prolog is and how it works. It also helps to explain the history of Prolog and its origins to provide the students with a holistic understanding of it.

Teaching Prolog for some years to art and design students, I observed the following: When I introduced Prolog as a programming language, the biggest problem students had with Prolog was the experience that the influence of their programming on the behavior of the program was limited. It is the experience that the system is doing something (namely: the resolution process including the depth-first search and unification) in order to generate output that the students did not program. Obviously, a part of the process is out of their coding control. Even more confusing: Not only do they have no control over the behavior, but they also have to adapt their whole coding to this black box, out-of-control behavior. And that is the obstacle that drives the confusion: The students don't understand why Prolog is called a programming language if they can't program the behavior of the system! It seems to them that they have to deal with two different things: There is a language AND an entity that reacts to the code that is written in that language.

This observation led me to redesign my teaching approach, introducing Prolog not as a programming language but as an AI, a technical entity that has a special inference behavior and that is equipped with a language in order to cooperate with that entity. I found out that this explanation (which goes back to the historic roots of Prolog in theorem-proving and Green's dream) [1],[4] gives the students a clear view of what Prolog is and how to use it. Looking at Prolog from the point of view that the language you use with Prolog is not a means to build a program from scratch, but that it is a means to communicate with an already existing, implemented program, namely an automated deduction system, makes it also easier to understand the difference between imperative and declarative programming as well as the difference between programming and meta-programming. Everything you do is in cooperation with the inference engine, including modifying its behaviour.

1.4.3. Reason No. 3: Making Prolog a perfect educational tool to shape digital, AI, and artistic literacy.

Prolog, understood as a logic-based AI, is an excellent educational tool for developing a wide range of fundamental skills. These skills are crucial for designers and artists in the digital landscape of the 21st century, and my teaching integrates their education within the overall concept of 'the reflective practitioner.' [8]:

- Shaping skills on how to handle ourselves as designers and artists around AI machines (for example: learning that there exists an alternative to subsymbolic AI in the form of symbolic AI)
- Shaping the skills for critical judgment of current technological developments in the fields of digitalization and AI with respect to historic, social, and aesthetic implications
- Shaping a sense for AI history (for example: understanding that the question-answer concept is much older than LLMs and that Prolog is a kind of early prototype of this concept and therefore a good tool to study it)
- Shaping interdisciplinary skills by learning to see the deep interrelations between mathematics, logic, computer science, computer art, and art
- Getting in touch with basic concepts of computer science and computing (concepts like database, formal language, grammars, set theory)
- Learning to understand the creation of an aesthetic object as a Form-finding problem and learning to look at a Form-finding problem from a logico-mathematical perspective
- Learning to describe an aesthetic object in terms of a formal language
- Shaping the understanding of the concept of compositionality and modularity on a fundamental level
- Shaping awareness of the universal use of concepts of modularity in everyday life (like architecture, bureaucracy, economics, games, industry)
- Shaping the understanding of logical thinking
- Shaping the skill to think, organize, and talk about the process of making an aesthetic object systematically, doing it in terms of logic
- Learning that using logical thinking is not a contradiction to creativity but an integral part of it
- Shaping the understanding of logic as a general methodology for problem-solving; learning to apply logic-based problem-solving strategies to creative tasks, and the extraordinary value it has to integrate the systematic use of it into one's own creative process
- Shaping the understanding of the difference between bottom-up and top-down approaches and learning how to use top-down strategies in the making of an aesthetic object systematically

- Understanding how using logic programming as the central tool in the generative process helps to understand, describe, and structure an artistic/design problem
- Shaping skills of creative coding
- Shaping the skill to put one's own creative use of programming into a broader historic context of art & design, seeing the connections of generative design with different reference points like:
  - The concept of Cybernetics
  - The Bauhaus School and Oskar Schlemmer's concept of "Kompositionelle Gestaltung" (Compositional Design) [5]
  - The Stuttgart School of Max Bense and his students Frieder Nake and Georg Nees pioneering the field of Computer Art on a sound theoretical basis with the concept of Generative Aesthetics, that later turned into what we know today as generative design [7]
  - The artistic strategy of ars combinatoria and its long history starting in ancient arts via the era of the Baroque to the concept of Montage and Collage and electronic art in the 20th and 21st century

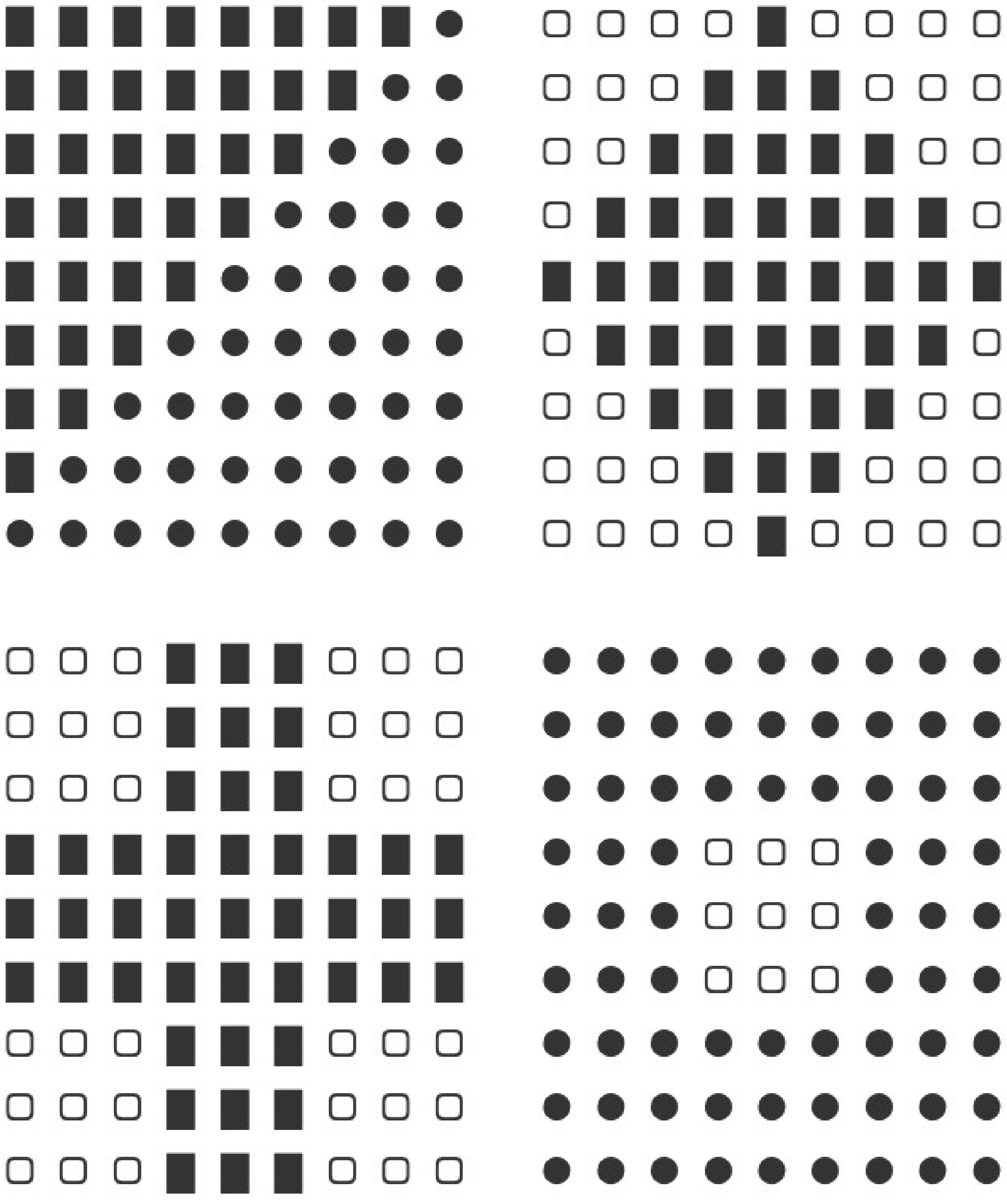


Figure 1: Output generated by a Prolog-program written by MA-student Alexander Kordes in the beginner's course Exploring Generative Logic at HSD, Summer Semester 2024

2. The method of Exploring Generative Logic

2.1. Introducing the students to Prolog as a logic AI

In the beginner's course, the students are introduced to the features of Prolog through the following interpretation of a standard Prolog System:

A Prolog-System consists of 3 components:

- An inference engine for automated deduction. That's the logic AI.
- A formal language, based on logic, used to communicate with that AI.
- Technical enhancements in the form of built-in predicates.

A classical Prolog-system is equipped with an inference engine that performs backward reasoning.

2.2. The task: Generating electronic mosaics

In the introductory course, we focus on generating a single class of aesthetic objects, which we call electronic mosaics. Electronic mosaics embody the central characteristics of classic mosaics: they are planar aesthetic objects composed of discrete elements.

This characteristic is also shared by many other aesthetic objects based on a rectangular grid pattern. Aesthetic objects from the fields of ancient art, Konkrete Kunst, ASCII art, typography, or visual poetry all share this feature. Rectangular grid patterns can be described straightforwardly in logic grammars using DCGs.

2.3. Conceptual point of departure: The concept of Generative Logic

The students are also provided with the basic viewpoint of Generative Logic:

According to the concept of Generative Logic, programming in Prolog means interacting with a logic AI. Interacting with Prolog means providing the logic AI with information about a problem that is true, and then asking the inference engine to draw conclusions from these initial statements of truth. Interaction with the inference engine happens in a formal language.

The inference engine then draws conclusions by performing controlled logical inferences based on the resolution principle discovered by J. Alan Robinson and further developed for practical use within a Prolog system by Robert Kowalski and Alain Colmerauer.

If the information provided by the user to the inference engine includes variables encoded within the program, then it is part of the proof-procedure that the inference engine binds the free variables to the corresponding values that it finds in the database. These bindings are given back by the system as output. This behavior, known as "Grounding Substitution" [2], is the key feature that makes Prolog a powerful generative AI.

Interesting to observe: The inference engine generates a version of the described object; the structure of the object remains invariant under the substitutions of its variables. It is possible to generate a multitude of variations using one single structure by mapping different values to each variable.

2.4. How do the students proceed?

Considering the conceptual point of departure described in 2.3, the students proceed as follows:

To generate an electronic mosaic, the students feed the inference engine with information about the rectangular grid pattern they want the engine to generate.

They proceed in 3 stages:

2.4.1. Stage 1: Defining a PUAO

In stage 1, they define what we call a PUAO: A partially unknown aesthetic object.

The students can define the following (basic) features of such a PUAO:

- the size
- the contour (i.e., the shape of the outline)
- the cellular structure of the PUAO

What do we mean by cellular structure?

The PUAO is understood as a complex of interconnected parts, a block of blank spaces, a composition of logic variables. “Variables allow us to state relationships among objects without naming specific objects” [3]. Therefore, each variable can be seen as an empty cell, a blank space reserved exclusively for a certain type of element that can occupy this cell. In other words, each variable designates a specific class of elements that can be bound to it by the inference engine.

The configuration of variables that defines the rectangular grid structure represents the spatial relationships between a set of different value classes in a plane. These values are potential elements of the electronic mosaic to be generated.

The description of the PUAO within the Prolog language takes place in two steps:

Step 1 Students describe the PUAO in pure Prolog by using predicates with free variables as arguments and integrating them into conditional language constructs. The use of built-in predicates is restricted to: `tab/1`, `write/1`, and `format/2`.

Step 2 Students adopt the DCG formalism and develop their program based on this approach.

The students can start with step 1 or the other other way round.

2.4.2. Stage 2: Constructing the database

In stage 2, the students construct the database: They decide what kinds of elements will be part of the database, define classes of elements, choose which element shall belong to which class, and define names for the predicates that designate class-membership of the elements having them as single arguments.

2.4.3. Stage 3: Developing a query strategy

In stage 3, the students develop a query strategy, thereby understanding the prompting of queries or compounds of queries as an important part of the process of composing an aesthetic object.

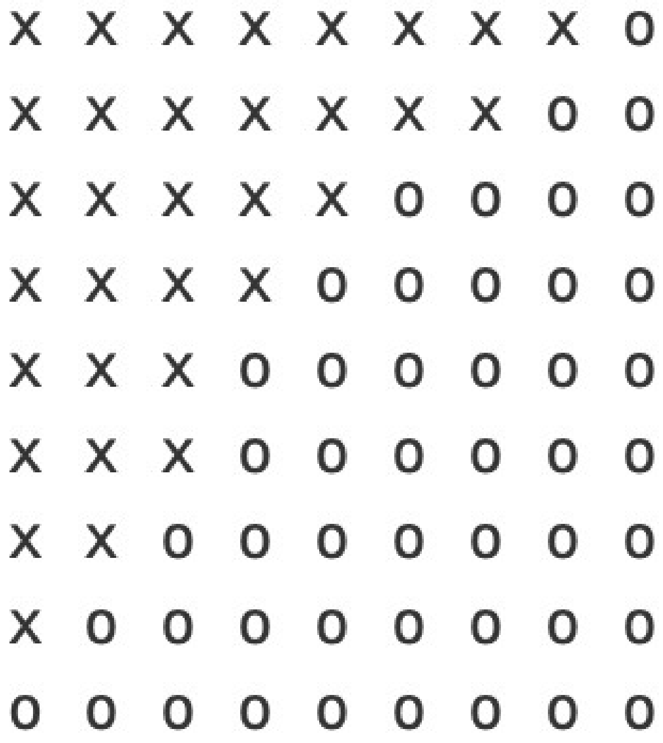
2.5. An example

What is shown here is a basic program in DCG notation used in the beginner’s course "Exploring Generative Logic" to describe a rectangular grid structure declaratively in a straightforward way. It is part of a set of basic programs currently being developed to tackle various aspects of programming in Prolog, including recursion. The program presented here consists of three sections as described above: the Partially Unknown Aesthetic Object, represented by non-terminals in the rules as a block of blank spaces; the sets of elements that fill these blank spaces, represented by terminals (note that these sets here are singleton sets); and finally, the query, which allows the use of the built-in predicate format/2.

Note that if you consider this code as a grammar, it differs from the standard usage of logic grammars. Instead of defining a set of objects, the code presented here defines a set of rules generating one single rectangular grid structure. The rules without terminal symbols remain consistent across all grammars in the family; however, the <nonterminal> –> <terminal> rules vary. This is where the terminals come into play. The output variety is achieved by replacing terminals in the body of these rules with other terminals, creating a vast field for experimentation. A program based on this concept was developed by M.A. student Alexander Kordes during the "Exploring Generative Logic" course at HSD in the summer semester of 2024. The output of that program is shown in Figure 01.

A basic program in DCG notation used in the beginner's course:

```
%Partially unknown aesthetic object:

q_01 --> z1,z2,z3,z4,z5,z6,z7,z8,z9.

z1 --> e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e2,n.
z2 --> e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e2,ws,e2,n.
z3 --> e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e1,ws,e2,ws,e2,ws,e2,n.
z4 --> e1,ws,e1,ws,e1,ws,e1,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,n.
z5 --> e1,ws,e1,ws,e1,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,n.
z6 --> e1,ws,e1,ws,e1,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,n.
z7 --> e1,ws,e1,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,n.
z8 --> e1,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,n.
z9 --> e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,ws,e2,n.


%sets of elements:
e1 --> [x].
e2 --> [o].

%white space:
ws --> [' '].

%next_line:
n --> ['\n'].


%query:
%nl,nl,nl, phrase(q_01, Ls), format("~s", [Ls]), nl,nl,nl,nl.
```



```
X X X X X X X X O
X X X X X X X O O
X X X X X O O O O
X X X X O O O O O
X X X O O O O O O
X X X O O O O O O
X X O O O O O O O
X O O O O O O O O
O O O O O O O O O
```

Figure 2: Output generated by the program shown above

2.6. What do the students learn?

Proceeding as described in 2.4, the students learn:

- To describe the known or required aesthetic features of the aesthetic object in a formal language based on logic
- How to set up a database in a concise and logical way
- To develop a concise, coherent query strategy (thereby learning the art of prompting)
- How to design all these three steps according to a well-thought-out plan on how to tackle the Form-finding problem using logical deduction based on a clearly envisioned intention

2.7. Two Sidesteps

2.7.1. Sidestep 1: Interesting to investigate: possible interpretations of what the students are doing

Learning to put their own creative action into different contexts of explanation, the students learn interdisciplinary thinking. Especially when they start to use DCGs, the students can look at their own creative action from different points of view: From a more computer-linguistic oriented point of view, the process could be seen like this:

Developing a PUAO, the students create a logic grammar for the composition of a planar aesthetic object from discrete elements, leaving the generation of the object to the inference engine. The students also select the repertoire of elements; in other words, they also compose the lexicon/alphabet that can be used in this process. The students create an artificial language to generate aesthetic objects.

From a more logical computer scientific point of view, the process could be described like this: The students provide the inference engine with a template, and the inference engine proves the satisfiability of this template by unifying its variables with the matching values found in the database, giving back ground instantiations of the template as output.

From a more general point of view, the process could be seen like this: The students develop a fill-out form and ask the logic AI to fill it out according to the given requirements stated in the form.

2.7.2. Sidestep 2: Generative Logic fosters interdisciplinary research between design and hard sciences

From an art & design point of view, along with the proceedings described in 2.4 come two very interesting fields of research and experimentation:

1. Finding (or developing) aesthetically interesting structuring instances to define the cellular structure of a PUAO.
2. Defining interesting classes of elements for the database. Finding and selecting interesting elements as members for each class.

These fields of research are predestined for interdisciplinary research because here we enter the domain of data visualization or sonification (for example, the visualization/sonification of biological, chemical, physical, or mathematical structures).

On the one hand, the method of Generative Logic can serve the needs of science in order to make complex structures, such as those investigated in biology or chemistry as well as in social sciences, etc., more accessible and easier to understand through visualization or sonification.

On the other hand, the method of Generative Logic is a perfect tool to explore the generative potential of such structures in order to generate aesthetically interesting output.

2.8. Outlook

Taking his own experiences with the students as empirical evidence into account, the author thinks that the EGL approach of teaching Prolog to beginners has a lot of potential that waits to be explored.

Among the possible domains to be systematically explored are these three:

2.8.1. Further development of the approach

Especially: Developing a collection of PUAOs as templates and inspirations for different courses, divided into categories that range from simple structures to complex and sophisticated ones.

2.8.2. Cooperations

The author assumes that the EGL-method can easily be adapted to other educational settings: Education of non-programmers of all ages, including children or students of other non-STEM curricula than art & design, in order to shape skills in logical thinking, digital, AI, and artistic literacy.

Therefore, the author thinks it is worth:

- Exploring if and how the approach could be adapted to other target groups
- Proving to what extent EGL can provide a useful complement to other interesting Prolog teaching approaches

2.8.3. Making the approach accessible

In order to provide other scholars who are interested in this approach, there needs to be developed:

- A toolbox along with a comprehensive handbook on how to teach Prolog the EGL way
- A workshop

2.8.4. Application of EGL to Different Artistic Domains

Further research could explore the application of EGL in various artistic domains, such as poetry and music. Over the past fifty years, Prolog has been widely used to generate art, poetry, and music, and there is a wealth of material available on this topic. It would be interesting to investigate how these approaches could be reintroduced within the conceptual framework of the teaching model proposed in this paper: making Prolog a gateway for students in art and design to enter the world of logic programming by teaching Prolog as a generative logic AI.

Acknowledgments

I would like to express my gratitude to the PEG 2.0 education group for their free sharing of ideas, which has been a great source of inspiration.

Special thanks to Veronica Dahl, Robert Kowalski, David Warren, François Fages, Michael Leuschel, Włodek Drabent, and Markus Triska for their extensive and enlightening discussions, which greatly contributed to shaping the ideas presented in this paper. I also extend my appreciation to my reviewers.

I am particularly grateful to Frieder Nake for the enjoyable and insightful conversations about the original concepts of generative aesthetics.

Lastly, I would like to thank Carsten Heisterkamp for assisting in refining the manuscript, as well as my students, who often take Prolog to unexpected places.

References

[1] Baron, Naomi. *Computer Languages, A guide for the perplexed.* London: Penguin Books, 1986.

[2] Flach, Peter. *Simply Logic.* Tilburg, 1994.

[3] Genesereth, Michael R. “Dynamic Logic Programming”. In: *Prolog - The Next 50 Years.* Ed. by Warren, D.S. et al. LNCS 13900. Springer, 2023, pp. 197–209.

[4] Hermenegildo, M., Morales, J., and Lopez-Garcia, P. “Some Thoughts on How to Teach Prolog”. In: *Prolog - The Next 50 Years.* Ed. by Warren, D.S. et al. LNCS 13900. Springer, 2023, pp. 107–123.

[5] Hüneke, Andreas, ed. *Oskar Schlemmer, Idealist der Form.* Leipzig: Reclam, 1990.

[6] Moles, Abraham A. *Kunst & Computer.* Cologne: Dumont, 1971.

[7] Nake, Frieder. *Aesthetik als Informationsverarbeitung.* Wien/New York: Springer, 1974.

[8] Schoen, Donald. *The reflective practitioner.* Basic Books, 1983.