Entwurf

Christian Jendreiko, Karl Svozil

10.09.2024

1 Unlocking the generative potential of quantum structures / partition logics using generative logic

1.1 Abstract

The paper is divided into two parts: In part one we consider our motivation and we describe what Partition Logics are and what the concept of generative logic is. In the second part we describe how the application of generative logic on partition logics results in interesting aesthetic output using a concrete example.

1.2 Key words

partition logics, logic programming, quantum structures, quantum music, quantum melodies, generative logic, complementarity, generative art, generative design, generative grammars

2 Basics

2.1 Reason Why / Motivation

The motivation behind this interdisciplinary project is the idea that mathematical structures can serve as structuring entities in order to generate interesting aesthetic objects. A well known example are Fractals. In this project we use Quantum structures that are represented in the form of partition logics as structuring entities for the generation of aesthetically interesting perceptual offerings. In order to do so we apply Generative logic on partiton logics.

The application of Generative Logic on Quantum logical structures in order to generate aesthetic output serves 2 objectives: on the one hand it explores in how far Generative Logic could be helpful to make quantum structures better understandable through visualization or sonification. On the other hand the project explores the generative potential of quantum structures in order to generate interesting aesthic output.

But what are partition logics and what is the concept of generative logic? In the next two chapters we will provide the reader with detailed descriptions.

2.2 Partition logics

Partition logics are similar to quantum logic in their ability to exhibit complementarity. Quantum complementarity is a fundamental concept in quantum mechanics that describes the relationship between 'properties'—or, more accurately, between various forms of perceiving—of a quantum 'object' that cannot be precisely known or measured at the same time. This is often referred to as a "trade-off" or "zero-sum game" between these properties. The most well-known example is the complementarity between position and momentum, where the more precisely you know the position of a particle, the less precisely you can know its momentum, and vice versa. This concept has far-reaching implications for our understanding of the behavior of particles at the atomic and subatomic level.

The kind of complementarity exhibited in partition logic derives itself from a (finite) set, which for pure convenience, is henceforth identified with some subset of natural numbers, beginning with the number 1 and increasing by increment 1 until a finite number n is reached.

For the sake of an example, we shall work with the set $\{1, 2, 3\}$.

A partition of this set satisfies two requirements:

(1) The set theoretic union of all elements of the partition is equal to the set being partitioned (in our case, $\{1, 2, 3\}$).

(2) All elements of the partition are mutually disjoint; that is, their set theoretic intersection is the empty set.

What are possible partitions of this set $\{1, 2, 3\}$? Well, there are three non-trivial partitions, namely $\{\{1\}, \{2, 3\}\}$ containing the two elements $\{1\}$, and $\{2, 3\}$, $\{\{1, 2\}, \{3\}\}$ containing the two elements $\{1, 2\}$, and $\{3\}$, and $\{\{1, 3\}, \{2\}\}$ containing the two elements $\{1, 3\}$, and $\{2\}$.

A partition logic is formed by

(1) representing a particular partition as a Boolean algebra, with the elements of that partition as atoms.

In our example, this would be the 4-element Boolean algebra formed by the $2^2 = 4$ elements $\{\emptyset, \{1\}, \{2, 3\}, \{1, 2, 3\}\}$, with the algebraic operations "and" and "or" identified with the set theoretic intersection and union. The logical implication is identified with the set theoretic subset relation, and the logical complement is identified with the set theoretic complement operation;

(2) a "pasting" (or stitching) of these Boolean algebras by identifying their common elements, analogous to creating a panorama from multiple photo views by stitching them together based on common elements, resulting in an enlarged set.

In our example, we could form a partition logic by pasting two or three partitions together. For instance, $\{\emptyset, \{1\}, \{2, 3\}, \{1, 2\}, \{3\}, \{1, 2, 3\}\}$ can be obtained by pasting $\{\emptyset, \{1\}, \{2, 3\}, \{1, 2, 3\}\}$ and $\{\emptyset, \{1, 2\}, \{3\}, \{1, 2, 3\}\}$.

Alternatively, we may consider $\{\emptyset, \{1\}, \{2,3\}, \{1,2\}, \{3\}, \{1,3\}, \{2\}, \{1,2,3\}\}$ obtained by pasting $\{\emptyset, \{1\}, \{2,3\}, \{1,2,3\}\}$, $\{\emptyset, \{1,2\}, \{3\}, \{1,2,3\}\}$, and $\{\emptyset, \{1,3\}, \{2\}, \{1,2,3\}\}$.

In this specific instance, there are only two common elements: the empty set $\emptyset$, which represents falsity or contradiction, and the original set $\{1,2,3\}$, which represents the tautology, being the antonym of the former. In this case, it is referred to as a "horizontal sum".

We use a certain class of partition logics that represent quantum structures. In the particular case discussed here as an example, the partition logics correspond to spin state measurements (by a Stern-Gerlach device) of a spin-1/2 particle along two and three measurement directions, respectively.

3 The Key Ideas of Generative Logic

Generative Logic (GL) is a conceptual framework for the application of the logic computer language Prolog* in the domain of art & design as a generative tool. It is being developed by one of the authors at the HSD.

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 & design as a powerful generative tool.
- GL understands logic AI as a white box, human-friendly, and trustworthy alternative/complement to black box subsymbolic AI.

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 about 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 there are variables involved in the information given by the user to the inference engine, 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" [**?**], is the key feature that makes Prolog a powerful generative AI.

Prolog, born around 1972, expresses a program as a set of Horn clauses. Computation is then performed by applying logical reasoning to that set of clauses. The approach was eventually described as "Algorithm=Logic+Control".

4 The Method of Generative Logic

How does the researcher proceed? Considering the conceptual point of departure described above, the researcher can proceed as follows:

Step 1:

In step 1, the researcher feeds the inference engine with a representation of the partition logic, interpreting it as a compound/composition of logic variables. "Variables allow us to state relationships among objects without naming specific objects." [**?**] Therefore, each variable can be understood as an empty cell, a blank space, that is reserved exclusively for a certain kind of element that can be placed in this cell. In other words: each variable designates a certain class of elements that can be bound by the inference engine to this variable.

The composition/arrangement/configuration of variables represents the spatial relations between a set of different classes of values in a plane. The values are possible elements of the electronic mosaic that is about to be generated.

The description of the partition logic within the Prolog language takes place within the DCG formalism of logic grammars. In Prolog, this formalism is called DCG. The abbreviation DCG stands for Definite Clause Grammars. This is a notation form in Prolog that was developed specifically for describing language grammars. DCGs can be used to describe character sequences abstractly and create them concretely. The alphabet used for this is divided into non-terminals and terminals; the basic syntactical construct used to form the individual character sequences has the form `head > body`.

Using DCGs, the NonTerminals are representing the variables and the Terminals represent the values.

Step 2:

In step 2, the researcher constructs the database: He decides what kind of elements will be part of the database, defines classes of elements, chooses which element shall belong to which class, and defines names for the predicates that designate class-membership of the elements having them as single arguments.

Step 3:

In step 3, the researcher develops 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.

Anm. (1)

See for example the explanations by Ulrich Neumerkel, in: Ulrich Neumerkel, Logikprogrammierung und Constraints Teil I, Skript, Technische Universitien, 2017, pp. 39-42: "Zur kompakten Beschreibung komplex strukturierter Listen gibt es in Prolog einen eigenen Formalismus. Durch eine Grammatik, auch DCG genannt (engl, definite clause grammar), werden Listen von Termen beschrieben." (Neumerkel, ebd. S.39) Eine solche Grammatik setzt sich aus einer Reihe von Regeln zusammen. "Eine Grammatikregel wird mittels des Pfeils (im Programmtext wie ¿,) angegeben." (ebd) "Grammatikregeln beschreiben eine Folge (Liste) von Termen. Das Komma wird und danach gelesen." (ebd. S.40) "DCGs und liche Formalismen werden in Prolog zur Realisierung von Grammatiken fr natrliche Sprachen und Computersprachen verwendet." (ebd. S.41)

5 From a quantum structure to an aesthetic object: A concrete example.

5.1 The representation of a V-logic in form of a partition logic represented as a logical grammar.

In the project the representation of a so-called V-logic encoded as partition logic is used to provide the structure for creating the character sequence of the non-terminals. A V-logic can be represented in different ways. One possible representation is the representation as a Greechie hypergraph. A Greechie hypergraph consists of two contexts with 3 nodes each. Because one of these nodes belongs to both contexts, the hypergraph consists of 5 nodes in total. The nodes are named a, b, c, d, e. The node c belongs to both contexts.

The graph can be represented as follows:
context 1: a'96b'96c
context 2: c'96d'96e
The two contexts as a combined Greechie hypergraph:
a'96b'96c'96d'96e

This graph can be used to describe the measurement results of a prototypical quantum physics experiment, for example a Stern-Gerlach experiment. In this experiment, the orientation of the angular momentum of a particle, the so-called spin, is measured along an axis. In one measurement, 3 measurement results are possible: spin up, spin down, no detectable result. (see Karl Svozil, Quantum Logic, 1998, p. 34)

The two contexts of the Greechie graph describe two different measurements within a series of measurements. The graph can be assigned a total of 5 states, in which truth values are assigned to the nodes. This is done according to the rule that in each context, only exactly one node out of 3 nodes can be assigned the value true, while the other two nodes can be assigned the value false.

This fact can be represented in form of a partition logic. In a partition logic each node is assigned two sets of states: The first set is the set of those assignments in which the respective node is assigned the value true. The second set is the set of all assignments in which the respective node is assigned the value false.

This results in the following partion logic:

a = {s1,s2}, {s3,s4,s5}. b = {s3,s4}, {s1,s2,s5}. c = {s5}, {s1,s2,s3,s4}. d = {s2,s4}, {s1,s3,s5}. e = {s1, s3}, {s2,s4,s5}.

Karl Svozil and I determined and noted this list of occupancy on September 11, 2023, in Vienna.

5.2 Transfering the partition logic into a Prolog program

The transfer of this partition logic into a Prolog program using the DCG notation is now very simple. All the researcher has to do is to apply the method of Generative Logic as described in 1.3.

5.2.1 Step 1

Applying step 1, the symbols / characters of the partition logic are translated into non-terminals. They are divided into two sets: The 5 nodes a,b,c,d,e become members of a set, we want to call set 1. The symbols of the partitions of the set of all states {s1,s2,s3,s4,s5} (s stands for state) become members of a set, we want to call set 2.

In addition to these two sets of Non-Terminals, we need two further sets of nonterminals, that are singleton sets: One singleton set with the nonterminal br and the other singleton set with the nonterminal n. The nonterminal br represents a separator between the two sets true and false of each partition. This non-terminal describes a separator that serves to optically separate the states. And the second non-terminal n denotes a line break. It is an important design element to enable not only a one-dimensional line arrangement but also a two-dimensional arrangement in which each node has its own line and these lines can be arranged one below the other.

Having defined these 4 sets of nonterminals, we are ready to construct the complete Non-Terminal structure.

To do this, the Non-Terminal structure is given a name. The individual nonterminals that are members of set 1 are assigned to this name as a sequence. In the prototype program shown here, the non-terminal structure is named v-logic. And the sequence of nonterminals associated with that name simply maps the hypergraph a'96b'96c'96d'96e:

$\mathrm{v}_l ogic'97 > a, b, c, d, e$

Next, to each member of set 1 are assigned the members of set 2 as a sequence, that represent the states in which the node represented by the member of set 1 is true followed by the nonterminal br, followed by the members of set 2 that represent the states in which the node represented by the member of set 1 is assigned the value false, followed by the nonterminal n:

Partition-Logic / v-Logic / Quantum Quadrat 1:

a –¿ s1,s2, br, s3,s4,s5,n. b –¿ s3,s4,br, s1,s2, s5,n. c –¿ s5, br, s1,s2,s3,s4,n. d –¿ s2,s4,br, s1,s3,s5,n. e –¿ s1,s3,br, s2,s4,s5.

Having this piece of code, Step 1 is accomplished. The composition of logic variables represented by the NonTerminals that are members of the four sets is established. Each of the non-terminals from set 2 can now be assigned a terminal, i.e. any symbol that ultimately appears on the screen.

We are ready for step 2 of the method: Constructing the database.

5.2.2 Step 2

As a proof of concept, within this project two different databases were developed that did serve as symbolic domains. One domain was developed for the Visualization of the Quantum structure. The other domain was developed for the Sonification of the quantum structure.

Both domains can be combined with the basic program that encodes the partition logic.

This opens up a wide field of design options. A deeper reflection on this topic will be found in 2.2.

In addition to the assignments of terminals to the nonterminals of set 1 and 2, the two non-terminals of the singleton sets are also assigned each one to a terminal: The separator: br –¿ [']. And the NonTerminal that is assigned to the terminal $\text{next}_l ine : n - - > [$
$n']$.

This creates the core program.

Having the core program we will move on to step 3 of the method: Developing a query strategy.

5.2.3 Step 3

In addition to this core program, there is another block of code that is necessary to output the realization described in the DCGs, but which plays no role in the design of the realization itself, apart from the placement of the realization on the screen in terms of the distance from the top edge of the output window and the distance of the realization from the word 'false', which always appears when the program is running in non-determination mode, i.e. all possible solutions are output. The non-determination mode is activated as soon as the format/2 function is used:

output :- nl,nl,nl, phrase($\text{v}_l ogic, Ls$), $format(" \ s", [Ls]), nl, nl, nl, nl.$

This code block can be understood in procedural terms as follows:

There is an output when the program produces three blank lines on the screen (nl,nl,nl) and (,) the inference machine replaces the non-terminals in the blank space structure v-logic with terminals and encodes this now filled blank space structure as a list and passes it on to the function format/2 (phrase($\text{v}_l ogic, Ls$)$and(,)the function format/$
$logic encoded as a list as a file type string(format(" \ s", [Ls]))and(,)the program then generates 4 blank lines(nl, nl, n$

If the program is now started by entering output in the query window, the following result appears:

5.2.4 A complete program

(85)

6 Exploring the generative potential of partition logics

Once a structure is encoded as s logic grammar, the grammar-representation of the partition logic can be mapped to any kind of symbolic domain we want, across all media boundaries. Via this way we can generate visual, textual or sonic patterns that do represent one and the same quantum structure. All that has to be done is to change the set of values that are mapped to the nonterminal structure.

In other words: Re-coding a partition logic as a sequence of nonterminals does open the gates into transmedia design.

Interesting point: The inference engine generates a version of the described object; the structure of the object remains invariant under the substitu- tions of its variables. It is possible to generate a multitude of variations using one single structure by mapping different values to each variable. You create variety when elements are changed Happy Accidents

(85)

7 Outlook:

The results show, that Partition logics offer a strong aesthetic potential as structuring entities that needs to be systematically explored. The Proof of concept does show also how easily it is possible to realise Qunatum Visualization or Quantum Sonification with the method of generative logic.

It is just a starting point. There is so much to explore. The selection of the according to which the Non-terminal structure is designed is at the discretion of the designer. This opens up a second, broad field of design options.

(85)