Constructive-Synthesizing Modelling of Multifractals Based on Multiconstructors

Viktor Shynkarenko [1,*,†], Robert Chyhir [1,*,†]

[1] *Ukrainian State University of Science and Technologies, Lazaryana str. 2, Dnipro, 49010, Ukraine*

Abstract

The concept of constructive-synthesising modelling is presented. The basic principles are determined. The classification of constructors by the purpose of constructing and external relations is presented. The types of constructors are defined: generating, transforming, analysing, optimising/adapting, algorithmic; autonomous, parametric, interactive, multiconstructors. Achievements in the application of the constructive-synthesising approach to solving a variety of problems are presented. The instrumental software "Constructor 1.1" has been developed. Simultaneously with the demonstration of the capabilities of the developed software environment, the formation of flat geometric multifractals is demonstrated. The formation of a multifractal is performed by realization a multiconstructor, which consists of a number of autonomous generating, parametric transforming and algorithmic constructors. The features of the deployment of transformations in the formation of constructors are shown: specialization, interpretation and concretization. The specialization of constructors determines the subject area of construction, necessary data and operations. To ensure the functioning of the constructive processes, all constructor operations must be interpreted by the corresponding procedures of the algorithmic constructor. The combination of the constructor (model of data and possible operations) with the algorithmic constructor (model of the executor) forms a constructive system capable of autonomous constructor by an internal executor. Substitution rules and initial conditions are setting in concretization. The developed software provides a certain flexibility in terms of possible modifications and the formation of new constructors and processes. This toolkit can be used as a basis for modelling various constructions and construction processes, especially in the tasks of their optimisation and structural adaptation.

Keywords

Constructive-synthesizing modeling, software, constructor, algorithm, formal grammar, formalization, fractal, information technology

1. Introduction

Consider the constructive paradigm of human perception of the world and the corresponding approach to programming.

The main principles of the constructive paradigm:

- the world is perceived as a set of constructions and constructive processes;
- constructions consist of some elements and other constructions;
- constructive process consists of elementary actions (discrete or continuous, deterministic or stochastic, etc.) and other processes;
- elements, constructions, intermediate forms of construction are interconnected by some relations of linkage;
- elements of constructions, intermediate forms of construction, constructions, relations and operations - each has its own set of certain attributes;
- the triad relation ⟶ operation ⟶ relation is used in the formation of constructions.

14th International Scientific and Practical Conference from Programming UkrPROG'2024, May 14-15, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

sinkarenko_vi@ua.fm (V. Shynkarenko); robertchigir@ukr.net (R. Chyhir)

0000-0001-8738-7225 (V. Shynkarenko); 0000-0002-7439-7368 (R. Chyhir)

The last statement requires some explanation. For example, in algebra of integers with a given addition operation, the entry «3+4» is perceived as a relation: you need to add 3 and 4, and the result of the addition operation is 7. In other words, we have a relation → operation: we need «to add» and «addition».

In the constructive paradigm: we have a relation «a screw can be screwed into a certain hole» (some of their attributes must match: the size of the hole and the bolt, the threading, the threading pitch). This relationship is the basis of the operation of screwing - a screw into a hole. As a result of the operation, a new screw to hole relationship is established. The relation between the hole and the screw has changed according to the triad of related relations and operations: can be screwed → screwing operation → screwed. As a result of the operation, all arguments (elements) with the new relationship remain.

The paradigm of constructive-synthesising modelling emerged as a result of an unsuccessful attempt to generalise the known tools of formal grammars. As it has been revealed, there are a large number of grammars' modifications [1, 2], such as multi-character [3], programmatic, stochastic [4, 5], weighted, index [6], matrix [7], various graphical [5] and grammar-like L-systems [8], R-systems [9], etc.

The main advantages of the proposed approach:

it is possible to link structures with constructive processes by using the same model;

optimise or adapt the structure of the construction and processes (components, their location, connections);

model constructions that are difficult or even impossible to obtain by other methods;

perform construction comparisons;

to model, analyse and predict constructive processes.

The paper considers the functionality of the software "Constructor 1.1" developed by the authors for automating the development of constructive-synthesising models and their use

2. Related works

A number of studies have been conducted using constructive-synthesising modelling, which demonstrate the productivity and effectiveness of this approach. There are some of them.

New possibilities for the formation of geometric fractals have been identified [10]. Namely, the formation of fractals from heterogeneous elements; combining different, including classical, fractals in multifractals. The possibilities of forming fractals have been expanded by eliminating the restrictions required by other approaches: the elements of the fractal formation must be represented by essentially non-intersecting sets; only compressive mappings are used. The new capabilities allow for new research and development in various fields such as theoretical physics or computer graphics.

The structural adaptation of algorithms and the data compression process is formalised by a set of three constructors [11]: a dual constructor of the compression algorithm (compression and decompression), a converter of the constructed algorithm into a constructive compression process, and the constructor-adapter. The constructors are implemented in the appropriate software, and its functional efficiency in compressing various data banks has been established. This made it possible to significantly increase the speed and efficiency of processing large amounts of information.

The developed constructive-synthesising model of natural language [12] allows: to consider language as a constructive process that can be used as a basis for creating a methodology for building systems with a high degree of intelligence; to improve the processes of semantic analysis, in particular, in the tasks of comparing and identifying the relevant semantic content in texts, thus significantly reducing the influence of synonyms, homonyms, periphrases and translation. The results of the constructor can be used to solve problems of developing intelligent systems for processing natural language texts.

Modelling crystal lattices for researching existing materials and modelling new materials, studying crystallisation processes. To model crystal lattices, constructive-synthesising modelling is used [13]. The constructors consider the attributes of the elemental base, which allows them to naturally build fractal spatial graphs according to the rules of substitution. Crystal lattices are viewed as special cases of fractal spatial graphs. This approach facilitates the creation of new materials with unique properties and allows for a different perspective on existing crystal shapes in the study and teaching of crystals.

The usage of constructive-synthesising modelling in the formation of lightning discharges based on the constructive-synthesising approach allows obtaining a realistic description of lightning frontal activity [14]. This approach can be the basis for solving the dynamic problem of lightning protection of engineering structures and civilian facilities, as well as developing a strategy for the behaviour of aircraft to minimise the risks of lightning strikes when moving in a thunderstorm front. This helps to improve flight safety and reduce risks during thunderstorm activity based on forecasting and analysing the activity of atmospheric masses during previous periods of activity.

The use of constructive-synthesising modelling tools has made it possible to formalise the processes and results of structure design at a logical level [15]. The constructor realization represents models of logically related specific data elements. A constructor model for representing and adapting data in RAM according to a logical model has also been developed.

Special software was developed to solve each of these and many other construction and modelling tasks (coding and debugging programs, hierarchy analysis, detecting borrowings in texts, determining the author's style and author of a text, fractal time series, university class schedules, and forming railway transport ontologies). The objective of this work is to simplify constructive modelling by developing a universal software. This software aims to combine various aspects of modelling and constructing, providing an integrated solution for a wide range of tasks.

3. Types of constructors

All constructors are formed on the basis of the generalised constructor, the theoretical foundations of which are laid down in [1] (where the outdated term for a constructor was used - formal structure).

A generalised constructor (abstract) is defined as:

$$C = \langle M, \Sigma, \Lambda \rangle, \quad (1)$$

where M is renewable heterogeneous carrier, Σ is signature of relations and related operations, Λ is information support for construction: purpose, conditions for starting and completing construction, substitution rules and restrictions....

The formation of real constructors is carried out by refining transformations of the generalised: concretization, specialization, interpretation and realization. Specialization defines the scope of use, its features, construction elements with attributes, and constraints. Interpretation matches all operations with algorithms for their execution, which can be performed by some internal performers. In the concretization, the rules for substitution and operations on attributes and initial values are set. The realization forms the resulting view of the constructor in a certain specified form. At the realization stage, the internal executor forms the constructor structures or performs constructive processes according to the specified rules and restrictions.

According to the purpose of use, constructors are divided into:

1. generating, which, according to the given rules, perform the formation of construction/constructive processes (one or more) within a certain described subject area;
2. transforming, which, based on one construction (its model), form another. For example, a program in processor codes into a constructive process of its execution;
3. analysing, which analyse constructors based on certain rules using specified methods. For example, to check whether the constructor can be formed according to the existing rules;

4. optimising/adapting, which are capable of changing the construction of a constructor or the constructive process (components, their order and connections);
5. algorithmic - in combination with other constructors, they create a constructive system: a model of elements, intermediate forms and possible operations on them is combined with a model of an internal performer who can perform all operations.

Constructors are distinguished by their external connections:

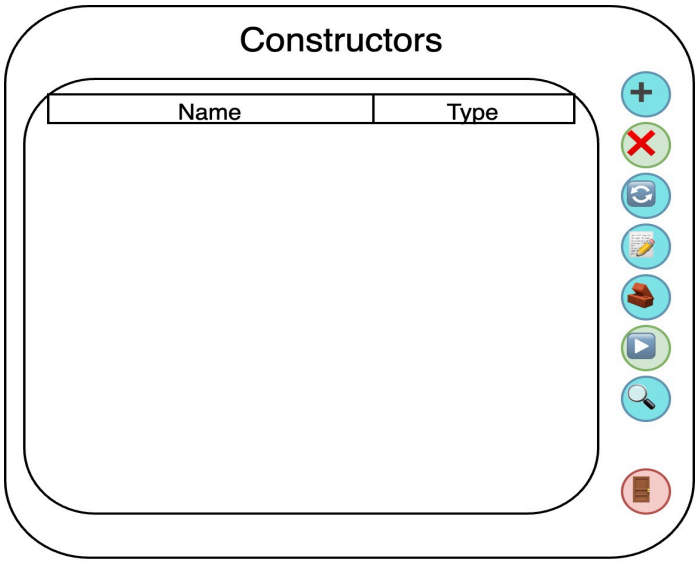
6. autonomous, which have all the necessary information to achieve the goal and do not require external information;
7. parametric, which are provided with external data before starting work through the parameter mechanism;
8. interactive, which have operations of obtaining data from the outside: from external media, including cloud, from the user, from external information collection devices;
9. multiconstructors that consist of several combined constructors based on the sequence of their realization. The constituent constructors are specified in advance for further combination in a multiconstructor.

4. Functionality of «Constructor 1.1»

The «Constructor 1.1" software was developed using Python with the usege of Qt technology to ensure cross-platform compatibility.

Demonstrate its functionality by modelling the plane fractal "Sierpinski Triangles" with the linear fractals "Koch's Snowflake" and "Sierpinski Curve" in the middle of the triangles.

To the list of constructors (Figure 1), add the constructors (Figure 2) one by one. Each of them



inherits all the properties of the generic constructor.

Figure 1: Constructors list.

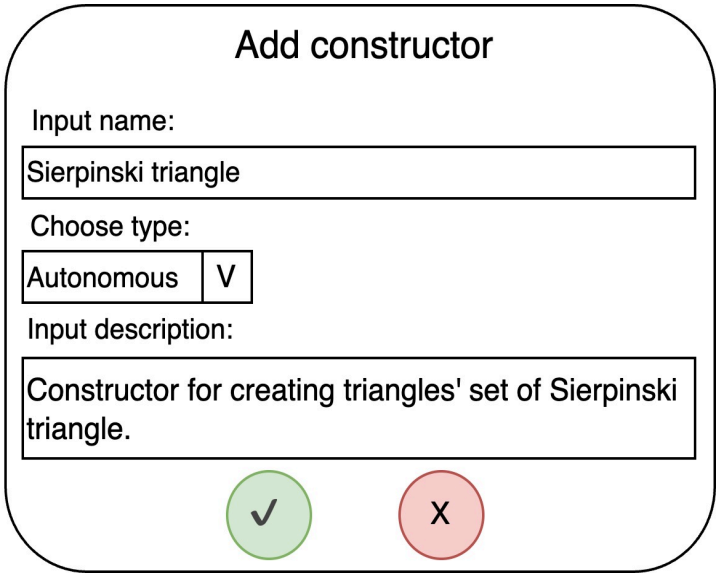


Figure 2: Adding a constructor with its identification.

The button menu on the form of Figure 1 contains the following buttons from top to bottom: add new, delete, duplicate, edit, refine transformations, realize constructor, display the multiconstructor diagram, and exit the program.

Add the first constructor.

When confirmed on the form of Figure 2, the autonomous constructor for generating the plane fractal "Sierpinski Triangle" is added to the list of constructors according to the form of Figure 1.

From the form in Figure 1, we proceed to the refinement transformations of this constructor (Figure 3).

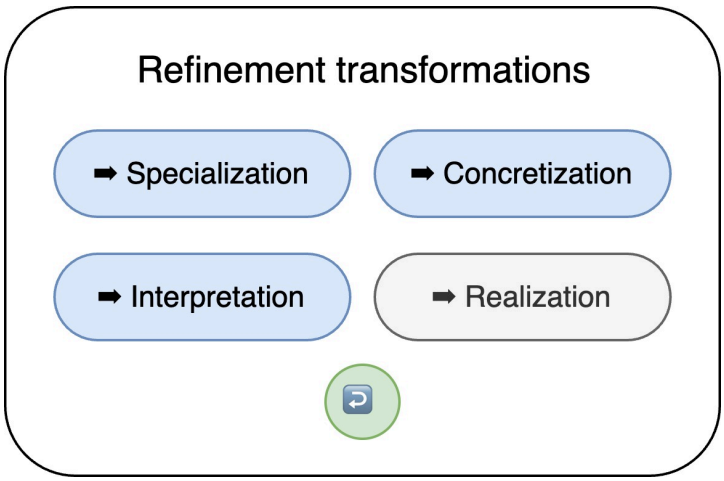


Figure 3: Refining transformations of the constructor.

In specialization, we define the main elements responsible for the preservation of triangles and set the purpose of each of them (Figure 4, Figure 5). Without specialization, interpretation, and concretization, realization is not available (impossible), but they can be specified in any sequence and a change in one may lead to a change in the other. Specialization identifies data with their types (Figure 4) and operations with a list of formal parameters (Figure 5).

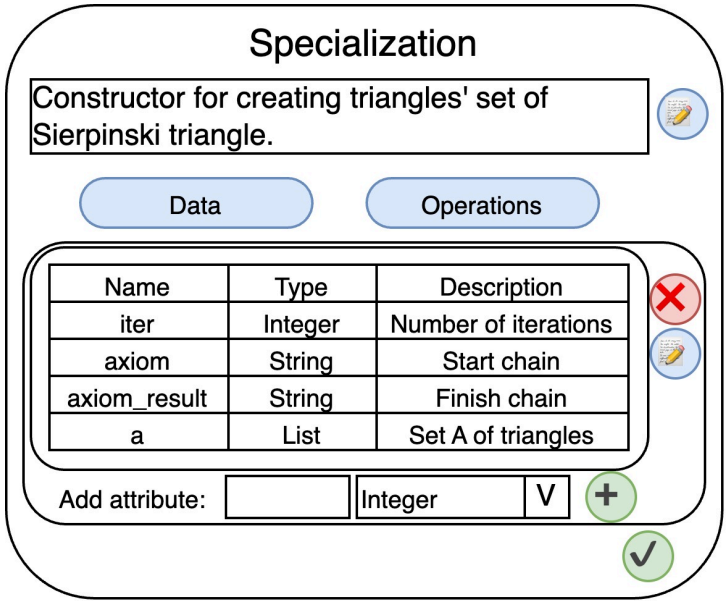
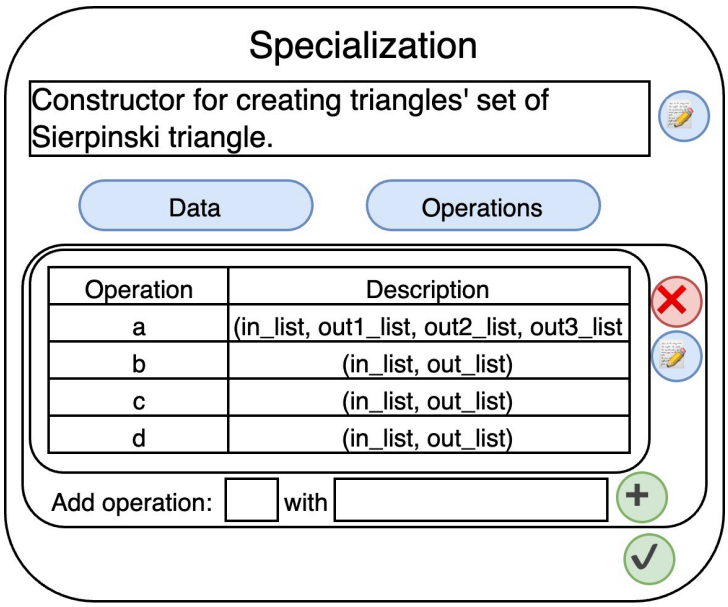


Figure 4: Specialization of the constructor: data.

In interpretation, we add the appropriate algorithmic constructor.

Define the basic algorithms in the algorithmic constructor "Triangular", the constructor is added to the list as in Figure 1. In specialization (Figure 6) is to provide well-established procedures that can perform all the operations of constructing a given plane fractal. Preparation and debugging of



these procedures is performed separately from this program in the Python environment.

Figure 5: Specialization of the constructor: operations.

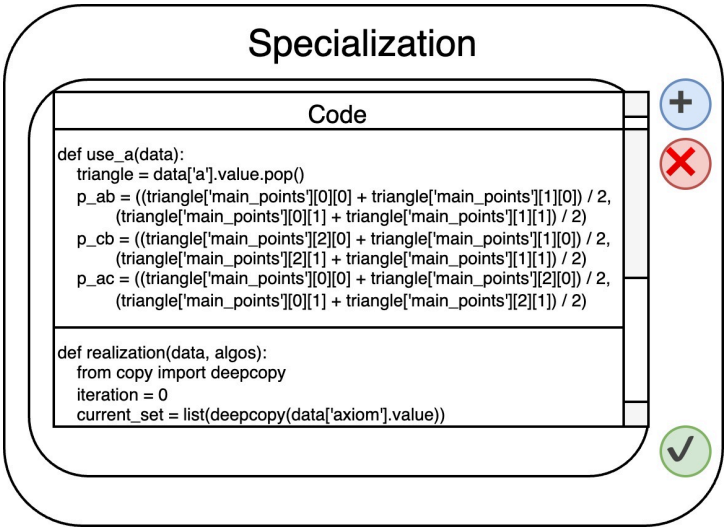


Figure 6: Specialization of the algorithmic constructor

This constructor is combined with a autonomous constructor and can be used not only in one constructor, since it performs the function of grouping algorithmic processes by a specific subject area. Changing the algorithmic constructor will lead to changes in all constructors where it is used.

After adding the algorithmic constructor "Creating triangles", return to the interpretation of the autonomous constructor "Sierpinski's triangle". On the form (Figure 7), we define the appropriate algorithmic constructor, add and match the operations in the specification of the "Sierpinski Triangle" constructor (Figure 3) with the algorithms of the "Triangular" algorithmic constructor (Figure 6) that can perform them. Algorithms that implement the operations of substitution, partial and full output are required [1]. As a result of the interpretation, a constructive system is formed that combines a constructor with elements and possible operations and a model of an internal performer capable of performing these operations.

The concretization sets the rules of substitution (construction) on the form shown in Figure 8, which consist of substitution relations and operations on attributes and initial conditions of formation (initial data values that were defined in the specialisation) on the form of Figure 9. The values are set according to the types of attributes. In this case, we set the initial number of triangles and their location on the coordinate plane in the corresponding attributes.

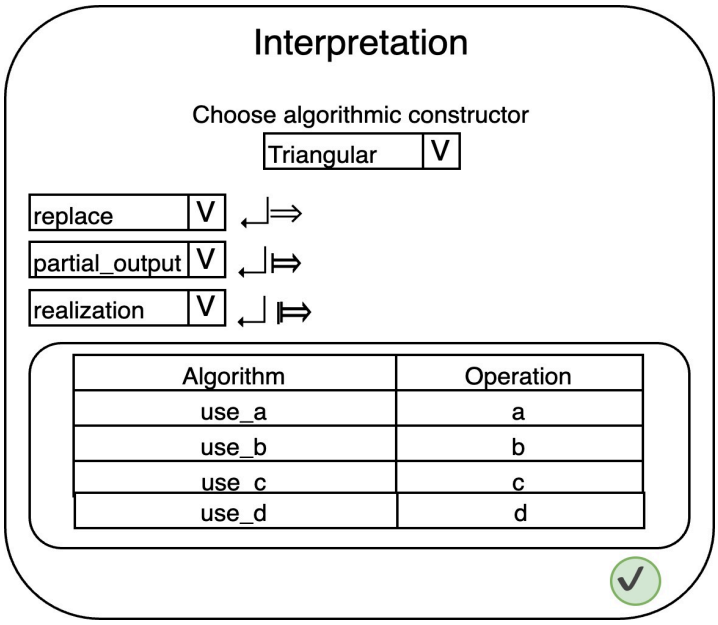


Figure 7: Interpretation of the constructor.

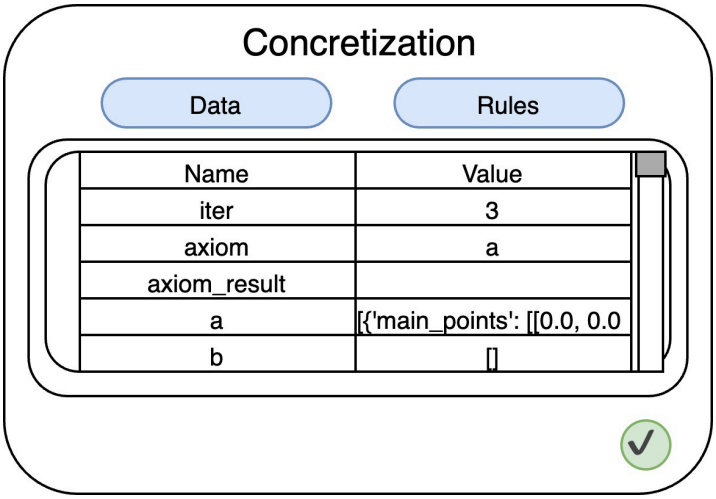


Figure 8: Setting data values in concretization.

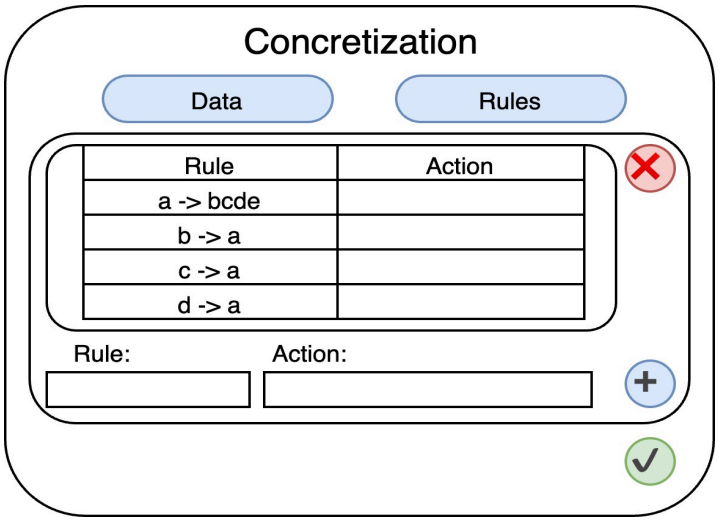


Figure 9: Adding substitution rules in concretization.

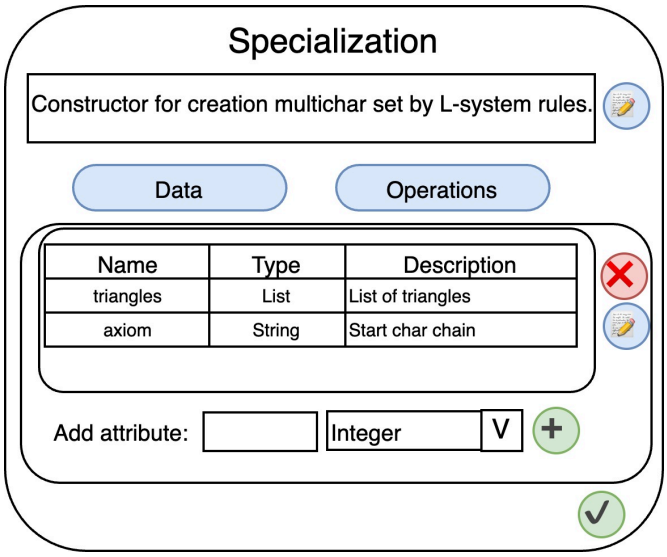
Since the constructor of the formation of a plane fractal is autonomous, after all the refinement transformations, it becomes possible to realize it, that is, to form a set of triangles that reflect the plane fractal "Sierpinski Triangle". The coordinate values of the shapes on the coordinate plane and their attributes will be saved to a specified file for further use. The formation of the data set is performed on the basis of the established rules and the specified algorithms in the constructors.

Next, each of the triangles in the resulting set is assigned a specific fractal shape such as a Koch snowflake or a Sierpinski curve. These fractals are formed on the basis of the corresponding reference values of L-systems.

Create parametric constructors for forming multi-character sequences for such fractals.

First, create a parametric multi-character constructor for the Koch snowflake.

In specialization, we will define the subject area of the constructors as the formation of fractals of this type and set the main attributes that are relevant for the formation of sequences based on the



input data set of the parametric constructor (Figure 10).

Figure 10: Specialization of the multi-character constructor.

For interpretation, we will add an algorithmic constructor with defined algorithms for generating multi-character sequences based on the rules of L-systems (Figure 11).

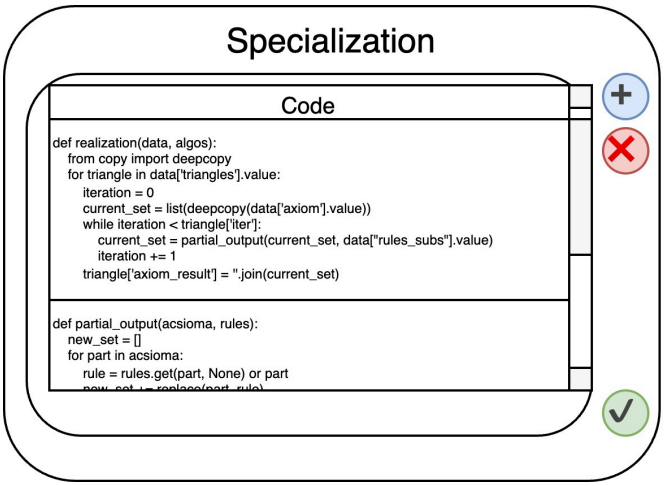


Figure 11: Specialization of the algorithmic constructor for generating multi-character sequence.

Link this algorithmic constructor to the parametric constructor "Koch's Snowflake L-system" to this parametric constructor (Figure 12).

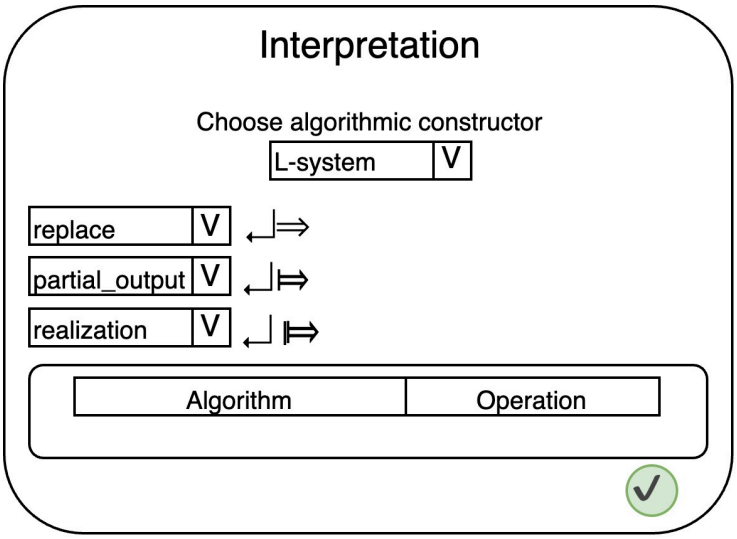


Figure 12: Interpretation of the multi-character constructor.

In concretization, define the initial data values (Figure 13) as the initial symbolic sequence and the substitution rules (Figure 14) for a particular fractal "Koch's Snowflake" (Figure 13).

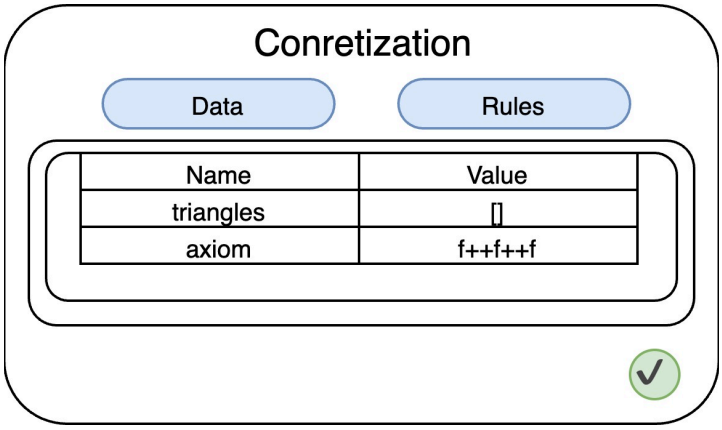


Figure 13: Setting data values in concretization of the multi-character constructor.

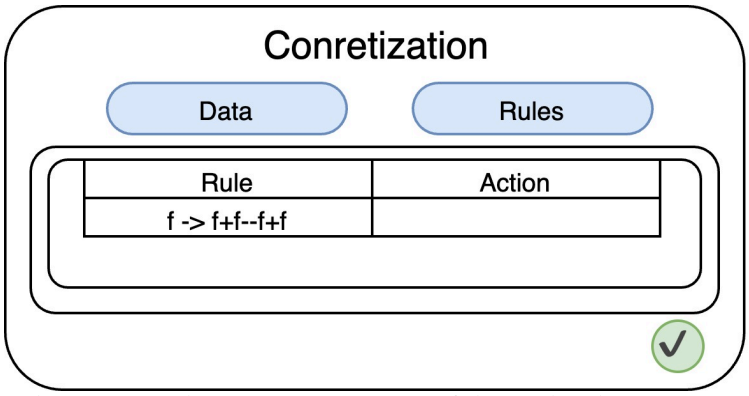


Figure 14: Adding substitution rules in concretization of the multi-character constructor.

In realization, this constructor will create multi-character sequences in each of the triangles that will be passed to it as parameters.

Similarly to the development of the "Koch's Snowflake L-system" constructor (Figures 10-14), we develop a constructor for the formation of the symbolic sequence "Sierpinski's Curve L-system".

In the same way as shown in Figure 10 - 14, we create parametric constructors for calculating coordinates for fractals based on the multi-character sequences "Koch's Snowflake Point" and "Sierpinski's Curve Point".

To graphically display the shapes formed as a result of realizations of the "Sierpinski Triangle", "Koch's Snowflake Point" and "Sierpinski Curve Point" constructors, we create a parametric graphical constructor to display the "Sierpinski Triangle" multifractal containing black triangles with the "Koch's Snowflake Point" fractal and white triangles with the "Sierpinski Curve Point" fractal, similarly to the previous constructors.

After defining all the autonomous and parametric constructors with the corresponding algorithmic constructors, we will combine them into one sequential process of forming a structure in the multiconstructor. Let's add the multi-constructor "Sierpinski Graphic Triangle" to the list of constructors.

At the stage of specialisation, we will determine the sequence of actions for creating constructors, forming structures and transferring data between constructors (Figure 15).

Figure 15: The order of realization the multi-character constructor.

In realization the multi-constructor, we will get a file with the image of the "Sierpinski Triangle" multifractal with the linear fractals "Koch's Snowflake" and "Sierpinski Curve" in triangles (Figure 16) at three iterations. The image corresponds to the appearance of the reference corresponding fractals.

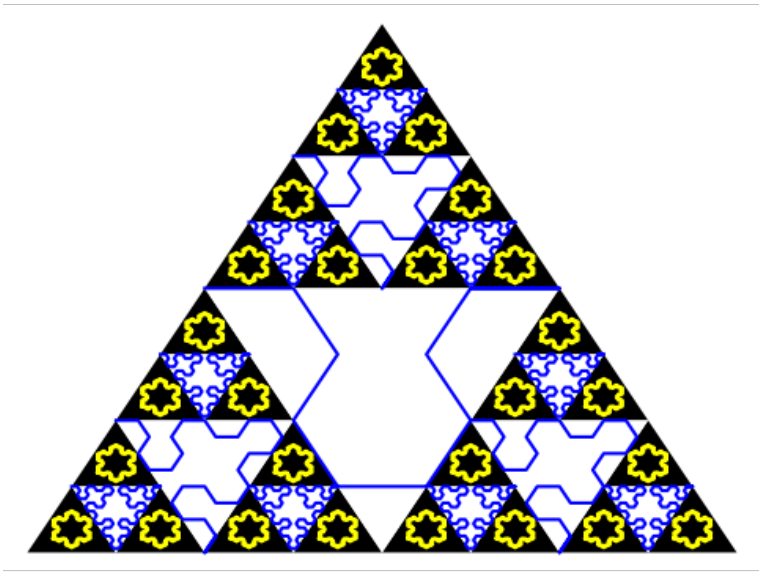


Figure 16: Sierpinski Triangle multifractal with Koch snowballs and Sierpinski curves.

5. Conclusions

The constructive-synthesising approach has been used repeatedly to confirm its relevance and to test it on the example of phenomena and processes from the world around us.

By developing constructors for other subject areas, it is possible to obtain constructions and constructive processes of a different nature. Each constructor is an independent and separate block in the formation of constructions.

The developed software can be the basis for modelling various constructions and constructive processes, especially in the tasks of their optimisation and structural adaptation.

The applied constructive-synthesising approach and the developed tools are flexible for the formation of multifractals. It can be modified to generate not only geometric and planar fractals, largely using ready-to-use software.

References

[1] V.I. Shynkarenko , V.M. Ilman, Constructive-Synthesizing Structures and Their Grammatical Interpretations.; Part I. Generalized Formal Constructive-Synthesizing Structure, in: Cybernetics and Systems Analysis, 2014. № 50(5), pp.665–662. doi: 10.1007/s10559-014-9655-z; Part II. Refining Transformations, in: Cybernetics and Systems Analysis, 2014. № 50(6). С.829–841. doi: 10.1007/s10559-014-9674-9.

[2] S. Müller, Grammatical theory: From transformational grammar to constraint-based approaches, Language Science Press, 2023. doi: 10.5281/zenodo.3992307.

[3] O. Iwashokun, A. Ade-Ibijola, Parsing of Research Documents into XML Using Formal Grammars, in: Applied Computational Intelligence and Soft Computing, 2024. doi: 10.1155/2024/6671359.

[4] R. Wandr, A.L.S. Ferreira, R.V.S. Pessoa, A. Galv, A. Machado-Lima, A stochastic grammar approach to mass classification in mammograms, in: IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2023.

[5] X. Pu, M. Kay, A probabilistic grammar of graphics, in: CHI Conference on Human Factors in Computing Systems, 2020. pp.1–13. doi: 10.1109/TCBB.2023.3247144.

[6] F. D'Alessandro, O.H. Ibarra, I. McQuillan, On finite-index indexed grammars and their restrictions, in: International Conference on Language and Automata Theory and Application, 2021. №279. pp.287–298. doi: 10.1016/j.ic.2020.104613.
[7] P. Prusinkiewicz, A. Lindenmayer, The algorithmic beauty of plants, Springer Science & Business Media, 2004. pp.228. doi: 10.1007/978-1-4613-8476-2.
[8] L.P. Lisovik, T.A. Karnaukh, A method of specification of fractal sets, in: Cybernetics and Systems Analysis, 2009. № 45(3). pp.365–372. doi: 10.1007/s10559-009-9117-1.
[9] V.I. Shynkarenko, Constructive-Synthesizing Representation of Geometric Fractals, in: Cybernetics and Systems Analysis, 2019. № 55. pp.186-199. doi: 10.1007/s10559-019-00123-w
[10] V.I. Shynkarenko, T.M. Vasetska, Modeling the Adaptation of Compression Algorithms by Means of Constructive-Synthesizing Structures, in: Cybernetics and Systems Analysis, 2015. № 51(6). pp.849–861. doi: 10.1007/s10559-015-9778-x.
[11] V. Shynkarenko, O. Kuropiatnyk, Constructive Model of the Natural Language, in: Acta Cybernetica, 2018. № 23(4). pp.995–1015. doi: 10.14232/actacyb.23.4.2018.2.
[12] V. Shynkarenko, O. Letuchyi, R. Chyhir, Constructive-synthesizing modeling of fractal crystal lattices, in: 18th IEEE International Conference on Computer Science and Information Technologies (CSIT), 2023. doi: 10.1109/CSIT61576.2023.10324251
[13] V. Shynkarenko, K. Lytvynenko , R. Chyhir , I. Nikitina, Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series, in: Advances in Intelligent Systems and Computing IV, 2020. № 1080. pp.173–185. doi: 10.1007/978-3-030-33695-0_13
[14] V.I. Shynkarenko, V.M. Ilman, G.V. Zabula, Constructive-synthesizing model of data structures at logical level, in: Problems of programming, 2014. № 2-3. pp. 10–16.
[15] J.W. Harris, H. Stocker, Handbook of Mathematics and Computational Science. Springer-Verlag, New York, 1998. pp.114–115.