Geometric Fractals' Constructive-Synthesizing Models using Ontological Means

Olena Kuropiatnyk [1,*,†], Viktor Shynkarenko [1,*,†], Larysa Zhuchyi [2,†] and Maria Lyakhova[1]

[1] Ukrainian State University of Science and Technologies 2, Lazaryana str., Dnipro, 49010, Ukraine
[2] railML.org e.V. Altplauen 19h 01187 Dresden, Germany

Abstract

The paradigm of constructive-synthesizing modelling is based on the idea of the world as a collection of different structures. The development of constructive-synthesizing modelling provides an opportunity to automate the formation of structures. Automation possibilities depend on the degree of formalization and the quality of the corresponding models. In this work, the formalization of constructive-synthesizing models is enriched by the ontological representation of knowledge. This approach is demonstrated in the formation and display of geometric fractals. The developed models are implemented by software tools using Java and Apache Jena framework. It is possible to change the basic elements of fractals based on their ontological representation.

Keywords

ontology, constructor, fractal, OWL, RDF, Apache Jena, constructive-synthesizing modelling, formal grammar, information technology, software

1. Introduction

One of the important stages of the software life cycle is design, namely, the development of a program model.

If software engineers, customers and users participate in the development of the model, then the models can be represented using standardized languages, such as XML, Entity Relationship Diagram (ERD), and others. They differ in the semantic richness. So far, several attempts have been made to analyze and organize them [16], [18]. Some of them have a long history of development and implementation e.g. UML. Their main advantage for domain experts is ease of use. But they always have to be combined with a software code or an expert, as they are not expressive enough to represent the semantics of the subject area.

Ontology is an explicit specification of conceptualization [13]. Ways of representing ontologies are languages OWL, RDF and SHACL. As their basis is description logic, it becomes possible to represent a greater number of rules and restrictions of the domain in the model itself, and not in the software code implementing it. There are light- and heavyweight ontologies. A lightweight ontology is a vocabulary that does not include rules and restrictions [9], that is, it does not make use of all the capabilities of ontological languages.

The authors previously modelled fractals using databases [20]. The application of ontologies is being researched for fractal modelling to clearly distinguish conceptual modelling and automation in software development.

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
olena.kuropiatnyk@ust.edu.ua (O. Kuropiatnyk); v.i.shynkarenko@ust.edu.ua (V. Shynkarenko); onto@common.railml.org (L. Zhuchyi); lyakhova.mariya@gmail.com (M. Lyakhova)
0000-0003-2286-884X (O. Kuropiatnyk); 0000-0001-8738-7225 (V. Shynkarenko); 0000-0002-9209-7262 (L. Zhuchyi); 0009-0002-3997-3304 (M. Lyakhova)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3909/Paper_33.pdf
CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

In this research the ontology is developed using the OWL Full language profile since the set of properties of data and objects overlap in the ontology.

2. Analysis of the application of ontologies in the modelling of fractals

We will adhere to the definition of a fractal as a set that has the property of self-similarity. These structures are used e.g. to develop maps in computer games [10] and to optimize storage and queries to big data [27], namely indexing, in which multidimensional data is transformed into one-dimensional data – reduces I/O costs, complexity of sorting and increases scalability.

Many tools designed for developing computer games use fractals to form fragments of background textures of the Earth's surface and the plant world. This allows for significant reducing the amount of data to be stored while increasing the realism of images.

To use fractals in software development, there is a need to somehow represent them. One such method is L-systems in JSXGRaph [11] – a Javascript library for rendering various functions, geometries and fractals in a web browser. For example Dragon curve representation includes rule "'l' : 'l+rF+'". The JSXGRaph library is currently actively used, for example, in statistical analysis [14], to visualize diagrams. The JSXGRaph graph is displayed on the screen using XML (HTML DOM).

XML is also used for serialization domain-specific language the Fractal Architecture Description Language (Fractal ADL) in the context of representing fractals [1]. Fractal ADL is the representation language of the Fractal component model (a domain-independent model of operating systems, graphical interfaces, etc.). The property of self-similarity is understood as "reflexive containment relationships" of system components [6]. The systems and their components are represented the same way at any level of abstraction [7]. The Fractal ADL language was developed 20 years ago but is still used in software development [15].

To represent the restrictions for Fractal ADL, add-ons are developed [8]: Fpath is a language of requests that can be used to check the reconfiguration of the computer system architecture, for example, the exclusion of cyclic dependence (a system component cannot include itself).

From the point of view of semantic richness, the next logical step after XML is UML and ontology models. Ontologies are actively used in the domains of transport, medicine, construction, etc. However, the formalization of fractal models using ontologies is an underexplored area. A simple example of using ontologies is the representation of fractals in RDF or JSON format. Here, RDF also allows checking by the Shapes Constraint Language (SHACL) the data for compliance with the developed Halcyon model – pathology imaging analysis and feature management system [4], as well as reusing well established ontologies like Annotations ontology [26].

Components of the "Fractal" system [1] can be combined with others, for example, components of the user interface and functions. In [3], this is implemented using the OWL ConcurTaskTrees ontology and SPARQL queries.

The GeoHilbert RDF dictionary [23], based on the GML, was developed to model spatial properties. Based on facts, queries are optimized using the same methods as in [27]. However, RDF vocabularies are lightweight ontologies and do not make use of description logic.

The presented analysis shows that fractal structures are used in information systems and software i.e. representation models (languages) are being developed for fractals. It was determined that there is a need for methods of representing constraints for fractal systems. A logical way to represent constraints is an ontology. But at the moment, they are not fully used in the domain of fractal systems.

One of the examples of the use of description logic in the domain related to fractal structures is the Collections ontology of data structures [5]. It makes full use of ontologies but includes concepts to represent simpler structures. In this work, a formal model of more complex fractal structures is developed.

In the end, it is worth mentioning that usually, developers are dissatisfied with OWL Full ontologies and try to "fix" them [24], i.e. convert them to OWL DL. This kind of correction

automation is implemented in e.g. Protégé reasoner Pellet [19]. However, some ontologies are intentionally developed specifically in the OWL Full profile [12], [17], [2].

3. Purpose

The purpose of the work is to improve constructive-synthesizing modelling by increasing formalization (enrichment of semantics) with the formal representation of knowledge by ontological means as soon as the more formalized the model, the more suitable it is for automating software development.

The work lays the foundation for such automation using the example of formal models of geometric fractals.

4. Constructive-synthesizing modelling of fractals

To formalize the process of forming fractals, we will use the apparatus of constructive-synthesizing modelling [21], [22], which is evolved formal grammars. To represent a fractal, we define a constructor and specialize it accordingly:

$$C = \langle \mathrm{M}, \Sigma, \Lambda \rangle \ {}_{S}\!\longmapsto, \langle \mathrm{M}_f, \Sigma_f, \Lambda_f \rangle, \quad (1)$$

where $\mathrm{M}_f \supset T$ is a non-homogeneous extensible medium containing a set of terminals T. Terminals include subsets of elements:

- graphic: basic, which are elements of construction, intermediate and final forms constructed on their basis, which have the property of self-similarity. Basic ones include lines, curves, and geometric shapes;
- symbolic: Latin letters indicating commands for the actor, which will be matched with graphic elements, signs of addition and subtraction operations – to indicate tilt angles; "→" substitution relation.

Σ_f is a signature of relations and operations performed on medium elements, Λ_f is a set of assertions of information support (ISC). ISC includes ontology, purpose, rules, constraints, initial conditions and conditions for end of construction.

4.1 Ontological model of the medium

Let's consider the ontology of the medium and represent it using OWL Full (Figure 1). The set of terminals includes elements of two classes that correspond to the graphic representation of fractals: basic Line_type and Fractal are the constructed elements that have the property of self-similarity. The set of attributes of a terminal is determined by its belonging to a class. Line_type has derived classes: Curve, Ellipse, Line, Rectangle, Triangle. The individuals of Line_type class can be described with the property has_width. Rectangle individuals can be described using the property has_height.

The set of constructed elements (fractals) is determined by their class. The Fractal class is basic. Its individuals can be described using the following object properties: available constants (hasConstant) and variables (hasVariable), line type (hasLineType), and starting terminal (hasStart). Their domain is individuals of the corresponding Constant, Variable, and Line_type ontology classes. The Fractal class individuals can be described with the following data properties: the tilt angle when constructing the fractal image, and the number of iterations during construction.

Subclasses of Fractal are Dragon_curve, Koch_curve, Sierpinsky_triangle. The subclass implicitly defines the structure of a fractal, which is described by the substitution rules $l_1 \rightarrow l_2$, where l_1, l_2 are forms that may contain terminal symbolic elements, "→" is the substitution relation. The Variable

and Constant ontology classes can be used to describe the terminal elements of the substitution relation.

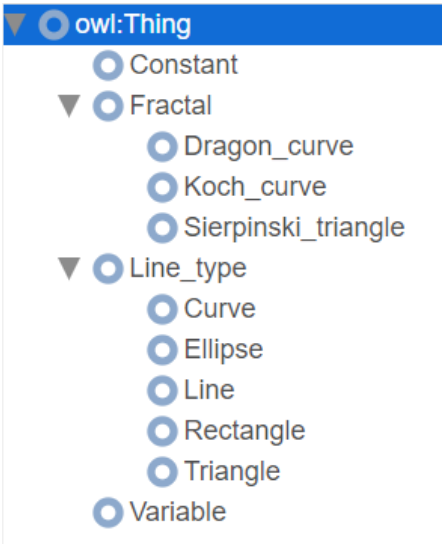


Figure 1: The class hierarchy of fractal description and auxiliary components

Data attributes of the Variable class are:

- hasMeaning_Koch_curve and hasMeaning_Sierpinsky_triangle – are description of actions done when constructing an image of the Koch curve and Sierpinsky triangle, respectively, if the current symbol in the sequence of symbols formed by the constructor is an individual of the Variable class;
- hasRule_Koch_curve and hasRule_Sierpinsky_triangle represent the right part of the substitution rule for constructing the image of the Koch curve and Sierpinsky triangle, respectively, the left part is an individual of the Variable class.

The data properties describing individuals of the Constant class are hasMeaning_Koch_curve and hasMeaning_Sierpinsky_triangle. They are the descriptions of actions when constructing the image of the Koch curve and Sierpinsky triangle, respectively, if the current symbol in the sequence of symbols formed by the constructor is an individual of the Constant class.

Some individuals of the Variable and Constant classes may have attributes that are not described in the class.

The medium ontology includes a basic set of individuals that is used as the values of object properties or used to construct new fractal individuals. Let's consider these individuals in more detail.

F1, F10 are individuals of the Koch_curve class. They are Koch curves, with different angles of inclination, the number of iterations and the type of line. The F1 individual is used as a base for forming other individuals. That is, new individuals of the Koch_curve class are built using data of the hasConstant, hasVariable, and hasStart properties of this individual. The structure of the F1 individual is shown in Figure 2. Rectangles with a purple contour are individuals and blue ones are ontology classes. Solid yellow lines are a class-subclass relationship. Dashed yellow lines indicate that individuals are of a certain class. So, for example, F is an individual of the Variable class, and F1 is a Koch_curve. Solid blue lines are properties. For example, the individual F is the value of the "hasVariable" property of the individual F1.

Individuals F2 and F3 are base for all new individuals of classes Sierpinsky_triangle and Dragon_curve, respectively. Their structure is like one of F1.

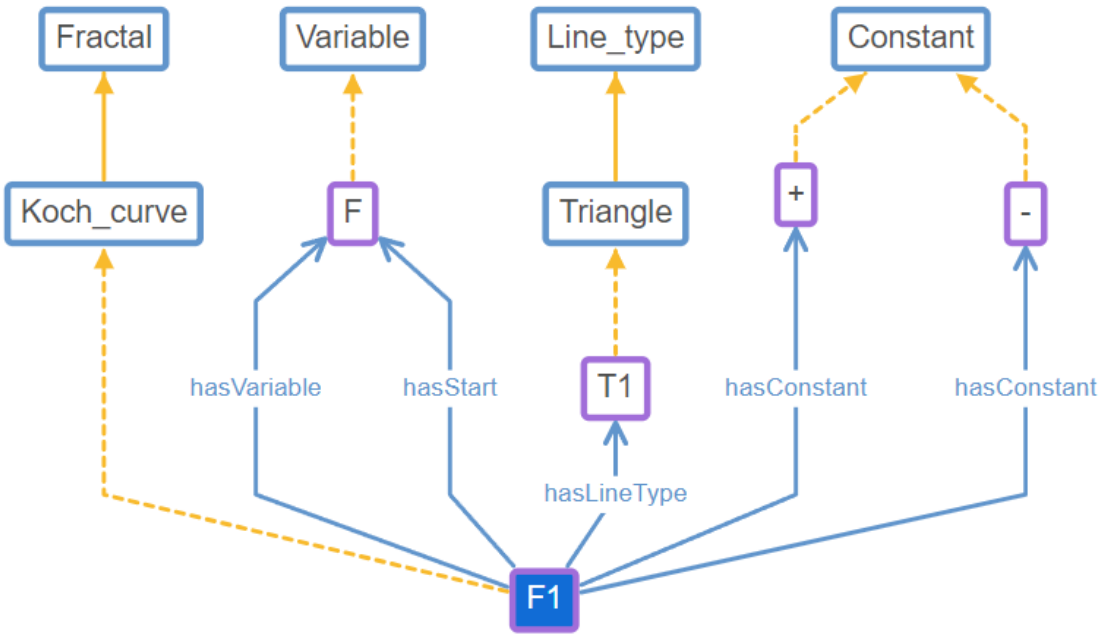


Figure 2: The structure of the Koch Curve individual

When forming a new individual of one of the Fractal subclasses based on individuals existing in the ontology, the following procedure is used:

1. the number of iterations and the angle are set as an integer,
2. line type is selected from the list of individuals of Line_Type subclasses,
3. set of variables and constants is copied from the base individual of the fractal of the selected class.

To set the line type, the ontology has individuals: C1 of the Curve class, E1 – Ellipse, L1 – Line, T1 – Triangle, and R1 – Rectangle. This list can be updated.

Individuals of the Constant class present in the ontology are "+", "-". They have data properties defined by the Constant class.

Individuals of the Variable class present in the ontology are F, and G. They have data properties defined by the Variable class. Individual F has additional properties:

- hasMeaning_Dragon_curve is description of the actions to be done during the construction of the dragon curve image, if this variable is processed in the rule;
- hasRule_Dragon_curve represents the right side of the substitution rule for drawing the dragon curve image, the left side is the given variable.

Such a difference in the attributes of the variables F and G is caused by the type of fractals they are used to construct.

Individuals of the Fractal class and its subclasses can have the hasStart property as a data property or an object property. This is due to the conceptualization of the property. If the derivation of the chain that describes the fractal according to the rules of substitution starts with a variable, then hasStart takes the value of the individual from the basic set of individuals of the ontology. If the output starts with a construction (chain) of variables and constants, then hasStart is represented by a string. So, for example, the derivation of the chain that describes the constructions of the fractal of the Koch_curve class begins with F – an individual present in the ontology. The derivation of the

chain that describes the fractal constructions of the Sierpinski_triangle class begins with F-G-G. Elements of the given chain are individuals present in the ontology.

4.2 Signature of operations for constructing fractals

Operations like concatenation, partial and complete derivation, as well as substitution relations, are defined on the medium elements.

The concatenation operation $\otimes(l_1, l_2)$ is the operation of linking form elements. If l_1 and l_2 are forms, then the form l_1, l_2 is called a concatenation of forms l_1, l_2. Any sequence of medium elements on which the binding operation has been performed will be called a form.

A substitution relation is a binary relation with attributes $l_i \ {}_{w_p}\!\rightarrow l_j$. For the form $l = \otimes(l_1, l_2, \dots, l_h, \dots, l_k)$ and an available substitution relation ${}_{w_p}(l_h, l_q)$ (can be written as $l_h \rightarrow l_q$) such that $l_h \prec l_q$ (l_h is part of l_q), the result of the ternary substitution operation $l^* =_{w_p}\!\Longrightarrow (l_h, l_q, l)$ will be of the form $l^* = \otimes(l_1, l_2, \dots, l_q, \dots, l_k)$.

Partial derivation operation ($| \Rightarrow (\Psi, {}_{w_i}\mathrm{l}_i)$), Ψ is the set of production rules consists of choosing one of the available substitution rules ψ_r: $\langle s_r, g_r \rangle \in \Psi$ with substitution relations s_r and performing substitution operations on its basis, g_r – operations on attributes, they are not used in this paper.

The binary operation of complete derivation or simple derivation ($|| \Rightarrow (\Psi, {}_{w_i}\mathrm{l}_i)$consists of a step-by-step transformation of forms, starting from the initial non-terminal and ending with a construction that satisfies the derivation termination condition, which involves the cyclic execution of partial derivation operations. The derivation termination condition is the completion of a certain number of cycle iterations.

The purpose of construction is to form a sequence of symbols for further transformation into a sequence of graphic primitives of the Line_type class, forming a geometric fractal.

Restrictions of the fractal constructor are set by the executor of the model, which affects the number and complexity of the constructed structures and the time of their formation.

The initial conditions for the construction of fractals are the presence of individuals of the Variable and Constant classes, which are used as the value of the hasStart property. The value of the has_n property is defined.

The construction completion condition is the execution of a specified number of iterations in the full derivation operation, which is determined by the value of the has_n attribute of the Fractal class.

4.3 Interpretation of the fractal constructor

We interpret the constructor using the algorithmic constructor C_A [21], [22]:

$$\langle C_f, C_{A,f} = \langle \mathrm{M}_{A,f}, V_{A,f}, \Sigma_{A,f}, \Lambda_{A,f} \rangle \rangle \ {}_{I}\!\longmapsto \ \langle \ {}_{A}C_f, \ {}_{A}C_f = \langle \mathrm{M}_1, \Sigma_1, \Lambda_1, Z \rangle \rangle \,, \tag{2}$$

where $\Lambda_1 \supset \Lambda_f$, $V_{A,f} = \{A_i^0|_{X_i}^{Y_i}\}$ is the set of basic algorithms, X_i, Y_i are the set of definitions and values of the algorithm $A_i^0|_{X_i}^{Y_i}$. $\Lambda_{A,f} = \{\mathrm{M}_{A,f} \supset \bigcup_{A_i^0 \in V_{A,f}} \left(X\left(A_i^0\right) \cup Y\left(A_i^0\right)\right) \cup \Omega(C_f)\}$ – non-homogeneous medium, $\Omega(C_f)$ is set of algorithms implemented by the C_f constructor; $\Lambda_1 = \{(A_1^0|_{A_i,A_j}^{A_i \cdot A_j} \downarrow \text{" } \cdot \text{ "}), (A_2 \,|_{l_i,l_j}^{l_i \cdot l_j} \downarrow \text{"}\otimes\text{"}), \left(A_3 \,|_{l_{h,l_q},f_i}^{f_i} \downarrow \Longrightarrow\right), \left(A_4 \,|_{f_i,\Psi}^{f_i \cdot} \downarrow |\Longrightarrow\right), \left(A_5 \,|_{\sigma,\Psi}^{\bar{\Omega}\cdot} \downarrow ||\Longrightarrow\right), \Lambda_1 \supset \Lambda_f$; Z is the set of possible executors (actors) that can implement algorithms A_i. The result of the actor's work is a constructed chain of symbols that describes the construction of fractals using the symbols of constants and variables.

The constructor ${}_{A}C_f$ includes algorithms for performing operations:

- $A_1^0|_{A_i,A_j}^{A_i \cdot A_j}$ – algorithm compositions, $A_i \cdot A_j$ – sequential execution of algorithm A_j after algorithm A_i;
- $A_2\ |_{l_i,l_j}^{l_i \cdot l_j}$ – concatenations of medium elements, l_i, l_j – forms;
- $A_3\ |_{l_{h,l_q},f_i}^{f_i}$ – substitutions, l_{h,l_q}, f_i – forms;
- $A_4\ |_{f_i,\Psi}^{f_i \cdot}$, $A_5\ |_{\sigma,\Psi}^{\overline{\Omega}\cdot}$ – partial and complete derivation, where $f_i f_j$ are forms, σ is an axiom, $\overline{\Omega}$ is a set of formed constructions.

By construction, we mean the form formed by multiple substitution and subtraction operations.

As a result of the interpretation, we get a constructive system, which consists of a construction model and a model of the internal executor.

4.4 The concretization of the fractal constructor

To clarify the entered operations, we will concretize the constructor C_A:

$$\langle_{I,C_A} C_{A,f} = \langle \mathrm{M}_1, \Sigma_1, \Lambda_1, Z\rangle\rangle\ {}_K\longmapsto\ \langle_{K,I,C_A} C_{A,f} = \langle \mathrm{M}_1, \Sigma_1, \Lambda_2, Z\rangle\rangle\,, \tag{3}$$

where $\Lambda_2 = \Lambda_1 \cup \Lambda_3$, $\Lambda_3 \supset \{M_1 \supset T, T = \{F, G, +, -\}\}$, is a set of terminals, F is the initial terminal for constructing fractals of the Koch_curve, Dragon_curve classes. For fractals of the Sierpinski triangle class, the FGG form is the initial terminal.

ISC: rules of substitutions. Let us consider the rules of substitutions, which allow us to formalize the process of building a chain of instructions for building a fractal image.

To construct a fractal of the Koch Curve class we apply the following rule:

$$\mathrm{F} \longrightarrow \mathrm{F} + \mathrm{F} - \mathrm{F} - \mathrm{F} + \mathrm{F} \tag{4}$$

where the symbols are commands: "-" – turn to the right at the specified angle (set as an attribute of the Fractal class); "+" – turn to the left by the specified angle, F – go forward and draw a graphic element (the type of the element is specified by the hasLineType property of the Fractal class).

To build a fractal of the Dragon's Curve class we apply the following rules:

$$\mathrm{F} \longrightarrow \mathrm{F} + \mathrm{G} \tag{5}$$

$$\mathrm{G} \longrightarrow \mathrm{F} - \mathrm{G} \tag{6}$$

where the symbols are commands: G – go up and draw a graphic element (the type of the element is specified by the hasLineType property of the Fractal class), the meaning of the other symbols is similar to Koch's curve.

To construct a fractal of the Sierpinski triangle class we apply the following rules:

$$\mathrm{G} \longrightarrow \mathrm{GG}, \tag{7}$$

$$\mathrm{F} \longrightarrow \mathrm{F} - \mathrm{G} + \mathrm{F} + \mathrm{G} - \mathrm{F}, \tag{8}$$

where symbols are commands with meanings given above.

5. Graphic representation of geometric fractals based on a constructive-production model with ontological knowledge

The developed constructor model (3) is the basis of the program for constructing images of geometric fractals. The program is written in Java using the Jena framework.

In the process of forming graphic representations of fractals, external and internal actors are involved (Figure 3).

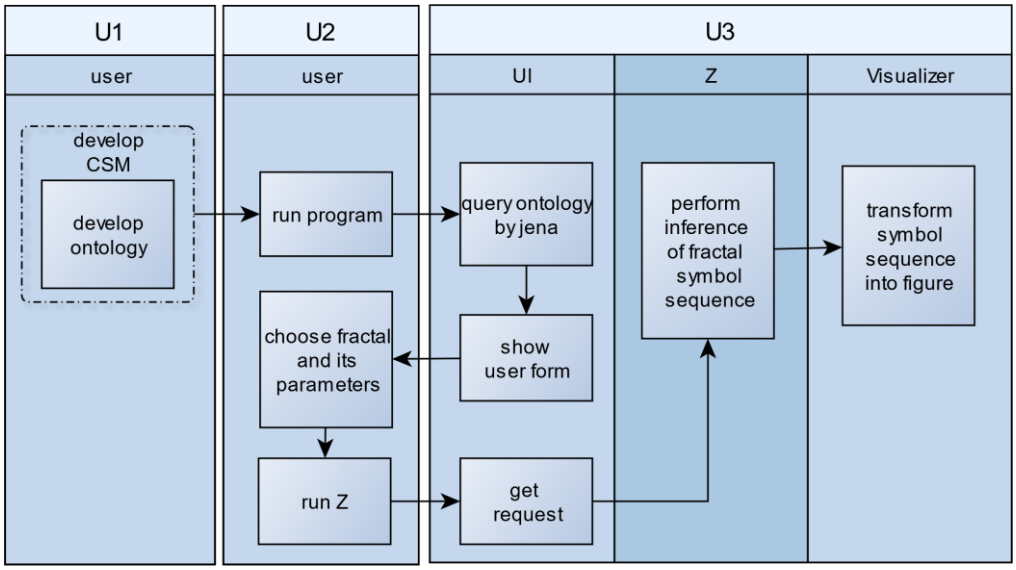


Figure 3: Workflow for formation and visualization of geometric fractal images

External actor:

- the developer of the constructive-synthesizing model including the ontology (U1);
- the user is the customer of constructions and the supplier of data for construction (U2);
- hardware-software system (U3), for the interaction of external actors: the user (U2) with Jena;
- a converter of chain structures (software and hardware, as in this case (part of U3), or an internal executor of another constructive-synthesizing model).

The internal actor (Z) forms fractal sequences of symbols based on the developed constructive-synthesizing model.

Let's consider the details of the internal actor. The implementation of the interpreted constructor ${}_{A}C_{f}$ consists of the formation of chains of symbols that satisfy the rules (4 – 8) by executing algorithms related to signature operations according to the rules of axiomatics:

$$ {}_{A}C_{f} \quad {}_{R}\longmapsto \overline{\Omega}({}_{A}C_{f}), \tag{9}$$

where $\overline{\Omega}({}_{A}C_{f}) \subset \Omega({}_{A}C_{f})$, $\overline{\Omega}$ is the set of formed chains of symbols that describe fractals, $\Omega({}_{A}C_{f})$ is the set of constructions that satisfy ${}_{A}C_{f}$ and can potentially be formed.

Let's consider the details of external actors. The formation of a graphic representation of a chain of symbols $\overline{\Omega}$ is performed by an external actor using the program. For this, it uses chains formed by the internal executor of the constructor (Z) according to the rules and parameters (number of iterations, angle), which are presented in the ontology. The construction of the chain is performed by separate methods (functions) of the program. The input data (medium elements) describing the fractal are read programmatically from the OWL file using a SPARQL query:

PREFIX f:<http://www.semanticweb.org/... # >

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject

WHERE { ?subject rdfs:subClassOf f:Fractal}.

Thus, the developed program represents an internal actor that implements the algorithms of the constructor (2), and an external one that uses the results of the work of the internal (chain of symbols) to form images.

For the elements, $\overline{\Omega}(\ {}_{A}C_{f})$ we will construct an image of a geometric fractal that corresponds to the description chain. 0 and 0 show images of basic fractals. These individuals are in the input data. The value of the properties of the individual F1 of the Koch curve is shown in the information panel of the program window on the left (Figure 4). Fractal F2, the graphic representation of which is shown in Figure 5a, has the following property values: angle – 120, number of iterations – 4, start terminal F – G – G, line type – ellipse.

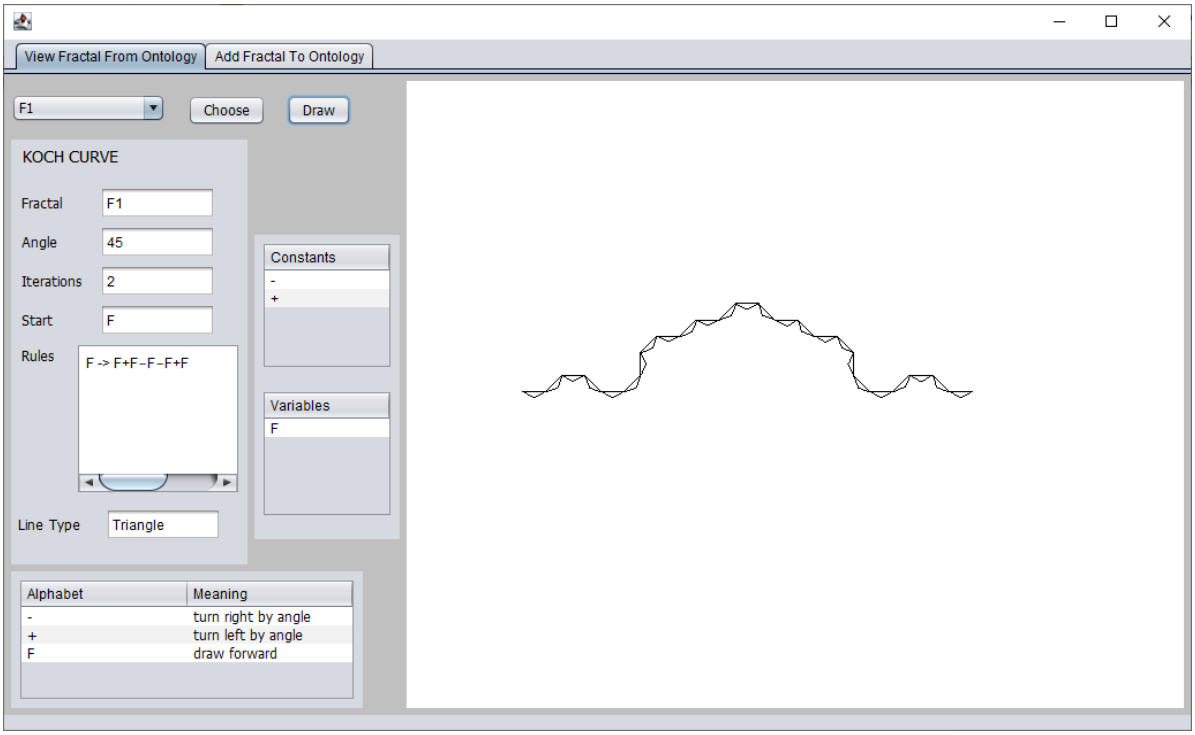


Figure 4: The main window of the program with a graphic representation of an individual of the Koch Curve class

Let's consider the construction of a graphic representation of the F2 individual in more detail. Rules (7) and (8) were applied for its construction.

The initial condition of vision is the string (axiom): F-G-G.

As a result of performing the substitution operation on each iteration (there are 4 iterations), we get a new chain (line).

To build a chain, in the first step, we choose a symbol to replace. Let's start with the first F. Apply the substitution rule F to it: F-G-G becomes (F – G + F + G – F)-G-G.

Step 2: choose the next F symbol to replace. We apply the substitution rule F to it: we get (F – G + (F – G + F + G – F) + G – F)-G-G.

Step 3: choose another F symbol to replace and apply the F replacement rule to it. The following steps are similar to the previous ones. As a result, we will get a chain of symbols, which is interpreted by the developed program as follows: F – command for drawing the graphic element horizontally, G is for vertically, plus and minus signs indicate the angles of rotation after the image by the F or G command. The type of the element is determined by an individual of one of the subclasses of the LineType class.

Fractal F3, the graphic representation of which is shown in Figure 5b, has the following property values: angle – 90, number of iterations – 8, start terminal S, line type – ellipse. All rules used during construction correspond to those described in section 4.4. These and other given images of geometric fractals were obtained analogously to the construction of the F2 fractal image.

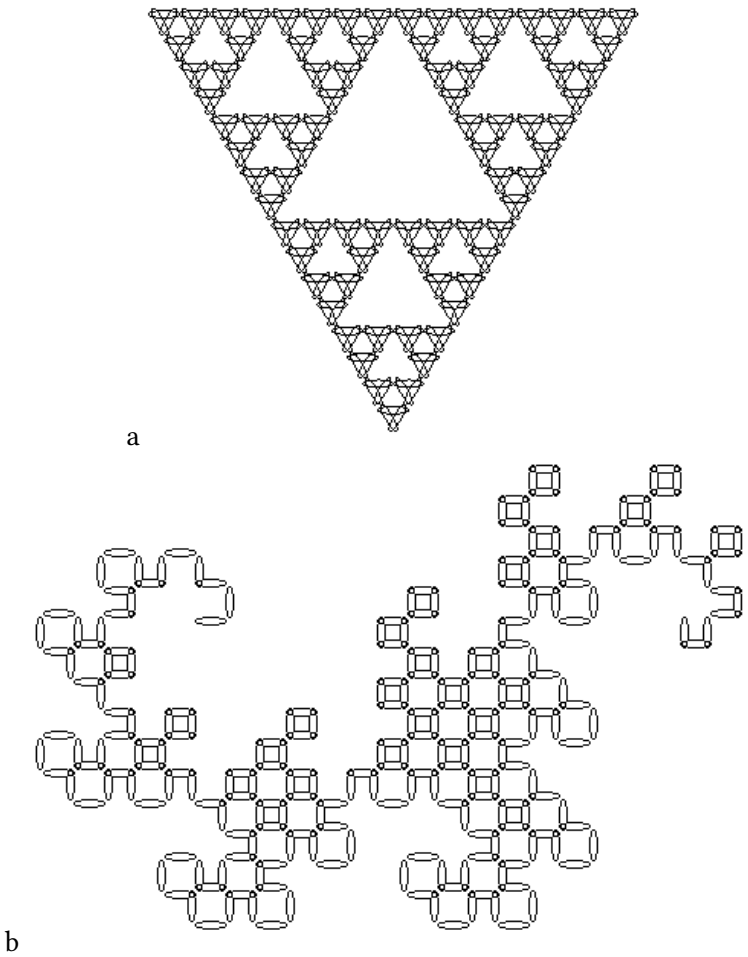


Figure 5: Basic fractals: a – Serpinski triangle, b – dragon curve

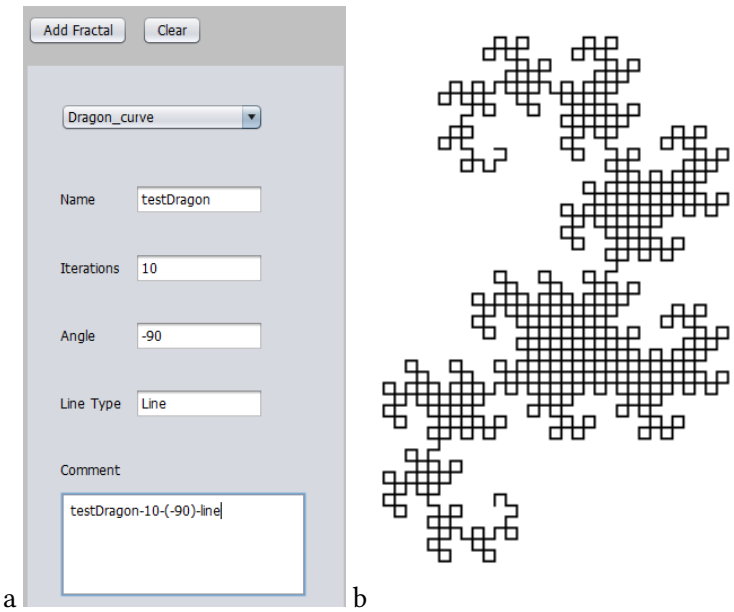
According to the scheme in Figure 3, the functionality of the program allows for the addition of new individuals, which is done by an external actor. An example of implementation is shown in Figure 6. Other examples of images of individuals added through the user interface are shown in Figure 7. The signature of drawing elements is given in the format "class - number of iterations - angle - line type".

The construction of fractal images for individuals, which are created during the operation of the program, is similar to basic fractals. Chain derivation uses a set of rules that was defined in the base fractal that was used as a basis for the current one. The number of iterations, angle and type of graphic element for new individuals is set by the user during creation. In the future, these parameters are used in the construction of the chain and its interpretation - construction of the image.

6. Discussion

The developed ontology of fractals (as part of the constructive-synthesizing model) can be classified as a lightweight ontology. The language of the ontology is OWL Full profile, that is, the descriptive logic was not used.

The definition of data types is a component of the model. Types impose restrictions on the set of valid values and operations on them. Such restrictions could be represented by constructs of the rdfs:domain type. Verification of the consistency of the schema and data can be performed by reasoners. Data types are currently checked by Java's built-in type conversion functions.



a b

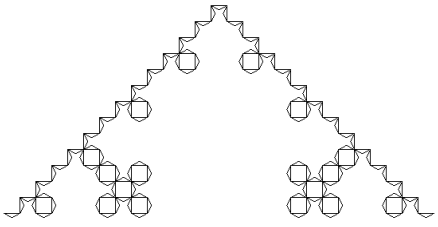
Figure 6: Creating a new individual: a – setting the parameters of a new individual, b – a graphic representation of the added individual

Setting fractal parameters and checking their correctness is carried out with the help of a graphical user interface implemented by Java tools. At the same time, there is a practice of validating data of similar forms by ontological means and is implemented using SHACL [25].

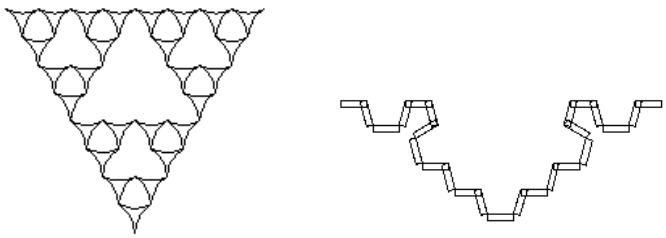
The properties of a new fractal are determined by copying them from existing individual fractals. In the future, to remove limitations of OWL Full, this can be implemented by OWL or SHACL SPARQL rules. In this case, they will be explicitly represented as part of the model.

In the future, the ontology of fractals can be extended by the presentation of other known fractals.

There are several alternative approaches to the formation of fractals: algorithmic, functional-algorithmic using a system of iterated functions based on a set of contracting mappings , L-systems, contracting affine automata.



a



b c

a – Koch curve-3 -90 -triangle , b – Serpinski triangle-3-120-curve ,
c – Koch curve-2-(-75)-rectangle

Figure 7: Elements of a set of implementations of the fractal constructor

Constructive-synthesizing modeling allows for creating a more types of fractals [20]. In particular, the possibilities of application are shown:

- large variation of color and forms attributes;
- non-uniform source elements of fractal formation;
- combining various, including classical, fractals in multifractals.

In this paper, the use of ontologies in constructive-synthesizing modeling extends the possibilities of generating fractals with a variable element base and initial conditions.

In the authors' opinion, the paper has achieved a twofold goal: providing a tool for working with fractals and a universal constructively synthesizing modeled structure based on ontologies.

7. Conclusions

As part of the pilot project, constructive-synthesizing models were improved with formal ontologies on the example of formalizing the processes of forming geometric fractals.

The results of separation of the automation of formation and conceptual modelling of fractals are given. It will be appropriate to enrich the ontology with rules and restrictions to ensure the verification of the consistency of the model by ontological means. For this, a further redistribution of the model constructions from the software code to the ontology must be performed.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] G. Blair, C. Thierry, S. Jean-Bernard, Component-based architecture: the Fractal initiative, Annals of telecommunications - annals des Telecommunications 64 (2009): 1-4.

[2] J. Borna, Ontology merging using semantically-defined merge criteria and OWL reasoning services: towards execution-time merging of multiple clinical workflows To handle comorbidities, Ph.D. thesis, Dalhousie University Halifax, Nova Scotia, December 2013

[3] C. Brel, A. M. Dery-Pinna, P. Renevier-Gonin, M. Riveill, Ontocompo: a tool to enhance application composition, in: Proceedings Human-Computer Interaction–INTERACT 2011: 13th IFIP TC 13 International Conference, Springer Berlin Heidelberg, Lisbon, Portugal, September Part IV 13, 2011, pp. 588-591.

[4] E. Bremer, T. DiPrima, J. Balsamo, J. Almeida, R. Gupta, J. Saltz, Halcyon--A Pathology Imaging and Feature analysis and Management System, arXiv preprint arXiv:2304.10612, 2023.

[5] P. Ciccarese, S. Peroni, The Collections Ontology: creating and handling collections in OWL 2 DL frameworks, Semantic Web Journal, 2013.

[6] T. Coupaye The Fractal Component Model, in: LAFMI Summer School, Puebla, Mexico, 2004. URL: https://fractal.ow2.io/doc/lafmi04/Fractal-LAFMI2004.pdf

[7] T. Coupaye, J.-B. Stefani. "Fractal Component-Based Software Engineering: Report on the WS Fractal at ECOOP'06." European Conference on Object-Oriented Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[8] P. C. David, T. Ledoux, M. Léger, T. Coupaye, FPath and FScript: Language support for navigation and reliable reconfiguration of Fractal architectures. Annals of telecommunications-annales des télécommunications, 64 (2009), 45-63.

[9] G. Fausto and I. Zaihrayeu, Lightweight ontologies, 2007.

[10] Fractal Mapper 9.0 NBOS Software. URL: https://www.nbos.com/products/fractal-mapper

[11] M. Gerhäuser, V. Bianca, and W. Alfred, JSXGraph-Dynamic Mathematics with JavaScript, International Journal for Technology in Mathematics Education 17.4 (2010).
[12] C. Golbreich, I. Horrocks, The OBO to OWL Mapping, Go to Owl 1.1!, OWLED, 258 (2007).
[13] T. R. Gruber, A translation approach to portable ontology specifications, Knowledge acquisition, 5.2, 1993, pp. 199-220.
[14] C. Hooper, I. Jones, Conceptual Statistical Assessment Using JSXGraph, International Journal of Emerging Technologies in Learning, 18.1, 2023.
[15] N.-T. Huynh, Towards Automatically Generating State Transfer Model Integrated in Adaptive Software Development Process, International Journal of Applied Engineering Research, 15.2, 2020, pp. 189-196.
[16] D. L. McGuinness, Ontologies come of age, in Fensel D., Wahlster W., Lieberman H. (Eds.). Spinning the semantic web: bringing the World Wide Web to its full potential, 2002, pp. 171-194.
[17] N. F. Noy, S. De Coronado, H. Solbrig, G. Fragoso, F.W. Hartel and M.A. Musen, Representing the NCI Thesaurus in OWL DL: Modeling tools help modeling languages, Applied ontology, 3(3), 2008, pp. 173-190.
[18] L. Obrst, Ontologies for semantically interoperable systems, in: Proceedings of the twelfth international conference on information and knowledge management, 2003.
[19] B. Parsia, S. Evren, Pellet: An owl dl reasoner, Third international semantic web conference, 18. 2004.
[20] V. I. Shynkarenko, Constructive-Synthesizing Representation of Geometric Fractals. Cybernetics and Systems Analysis, 55(2), 2019, pp. 186-199.
[21] V. Skalozub, V. Ilman, V. Shynkarenko, Development of Ontological Support of Constructive-synthesizing Modeling of Information Systems, Eastern-European Journal of Enterprise Technologies, 6(4), 2017, pp. 58-69.
[22] V. Skalozub, V. Ilman, V. Shynkarenko, Ontological Support Formation for Constructive-synthesizing Modeling of Information Systems Development Processes. Eastern-European Journal of Enterprise Technologies, 5(4), 2018, 55-63.
[23] C.-J. Wang, K. Wei-Shinn, and C. Haiquan, Geo-store: a spatially-augmented SPARQL query evaluation system, in: Proceedings of the 20th International Conference on advances in geographic information systems, 2012.
[24] T. D. Wang, Gauging ontologies and schemas by numbers, in: Vrandečić, D., del Carmen Suárez-Figueroa, M., Gangemi, A., Sure, Y. (eds.) in: Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web (EON2006) at the 15th International World Wide Web Conference (WWW 2006), Edinburgh, Scotland. 2006.
[25] J. Wright, S. J. Rodríguez Méndez, A. Haller., K. Taylor, P. G. Omran, Schimatos: a SHACL-based web-form generator for knowledge graph editing, International Semantic Web Conference, Cham: Springer International Publishing, 2020, pp. 65-80.
[26] Web Annotation Data Model. W3C Recommendation 23 February 2017. URL: https://www.w3.org/TR/annotation-model/ Date access: 11.13.2024
[27] X. Zhang, L. Chen, Y. Tong, M. Wang, EAGRE: Towards scalable I/O efficient SPARQL query evaluation on the cloud, 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 565-576.