# Formal Grammars Generating Fractal Descriptions of Molecular Structures

**Savino Longo**
*Dipartimento di Chimica*
*Università degli Studi di Bari Aldo Moro*
*Via Orabona 4-70125, Bari, Italy*
*and*
*Istituto per la Scienza e Tecnologia dei Plasmi*
*Consiglio Nazionale delle Ricerche*
*Bari Section - Via Amendola 122/D - 70125, Bari, Italy*

Simple rewriting rules are used to produce alphanumeric strings that embed fractal number sequences and are directly translatable into descriptions of hydrocarbon structures of considerable complexity, featuring hierarchical schemes. Rotations of the alphanumeric strings lead to radical rearrangements of the corresponding structures, which lose their initial schemes and become much less predictable, featuring different topologies of polygonal cycles. This shows that a complex and not necessarily ordered molecular structure may nevertheless have a relatively low algorithmic complexity. The variety and versatility of reorganization in chemical topology, due to the nonlocal representation of bonds in the coding string, may have played a role in prebiotic chemistry.

*Keywords*: molecular modeling; formal languages; generative grammars; polycyclic alkanes; L-systems; algorithmic complexity; fractals

## 1. Introduction

It is reasonable to think that a complex molecular structure requires a complex description. To confirm this, consider the task of attributing it a name, formulated according to standardized rules [1]. From an information theory point of view, this is equivalent to assuming that a complex chemical structure must have a high algorithmic complexity, in the sense of Kolmogorov and Chaitin [2–3]. Useful tools for such studies are provided by the formalized languages developed in the last few decades. These languages are used to feed input into programs that store and communicate chemical structures in a very efficient way. A well-known example is the SMILES language [4]: a SMILES expression provides a complete description of a chemical structure in every respect apart from hydrogen bonding, supra-molecular arrangements or topology. Initially, we might assume that a complex

https://doi.org/10.25088/ComplexSystems.32.4.395

molecule requires a long SMILES string, and therefore proportionally more bits of information. However, information theory points in another direction, suggesting that a linear expression could in some way be compressed significantly, suggesting low complexity. A recent development in this direction is assembly theory (AT), which aims at characterizing the complexity of a molecular system in terms of the steps necessary to build it recursively, starting from smaller fragments. The number of such steps is called the "assembly index" and has been proposed as a measure of complexity, with the ambition of discriminating molecules of biological origin from abiotic molecules [5].

At the same time, the capability of mathematical schemes based on the repeated application of simple substitution rules in formal expressions to produce structures of remarkable complexity has been equally known for many decades. These schemes are often difficult to connect to the generative algorithm employed in retrospect [6]. Examples are the deterministic fractals obtained by systematic, iterated application of geometric replacement rules [7].

Hence the idea arises that algorithms of this type can produce atom-by-atom descriptions of complex chemical systems, to be used for subsequent elaborations and characterized in a complete way by simple, deterministic programs.

In this paper, we elaborate this point of view. As a concrete example, we consider the generation of alphanumeric strings through the iteration of systematic syntactic substitutions to describe polycyclic hydrocarbons. It is shown that the result of these iterations, from the point of view of the final molecular structure, quickly escapes predictability and requires detailed analysis.

## 2. Unrestricted L-systems

A solid base for the efficient generation of molecular models is provided by L-systems [8], as well characterized in the field of descriptive biology [9, 10].

L-systems are examples of rewriting systems, algorithms for elaborating strings. In addition, in an L-system the string generated is interpreted as a set of instructions, and as such used by a virtual machine that interprets it to produce images.

An L-system can be described as a multi-component structure $(V, \omega, P)$ where $V$ is a set of symbols that allows the construction of words, $\omega$ is a starting word or axiom, and $P$ is a production operator that associates a new word to an argument word by replacing symbols in a systematic way.

$P(X)$ makes replacements in $X$ of the type $A \rightarrow B$, where $A$ and $B$ are strings of symbols.

L-systems easily produce patterns with branched fractal structures: this possibility, much exploited in biological simulations, has been applied to chemical structures in a recent paper by the author [11] to show that the applications of L-systems to molecular expressions produce models of complex branched structures. To this aim, the representation of branching by parentheses was exploited, as in a language like SMILES.

However, the way L-systems are normally defined produces structures with behavior that is somewhat predictable: this derives from the fact that they are based on systematic modifications and that they do not account for the new context produced by previous modifications of the string. They are therefore part of the so-called context-free grammars, ranking low in Chomsky's hierarchy of generative grammars [6].

It is therefore promising to explore chemical descriptions generated using linguistic operators that, while still deterministic, overcome this limitation.

To this aim, the operator $P$ in $(V, \omega, P)$ must be replaced by a more generic, unrestricted one $\Pi$, whose operation is affected by the whole structure of its argument $X$, for example, by counting the number of symbols in $X$. This produces a more general grammar $(V, \omega, \Pi)$, ranking higher in Chomsky's hierarchy.

In this paper, we use operators that exploit the conventional meaning of pairs of numbers in SMILES: to indicate the bond between two non-contiguous atoms in a string describing a structure. The possibility of creating bonds between atoms that are at an arbitrary distance in the string expression allows us to create, using generative grammars, models of systems that are not only polycyclic, but organized in a complex and hierarchical way. We discuss the symbolic dynamics brought about by the repeated application of these operators.

## 3. Linguistic Operators

In this paper, as in [11] and [12], we use linguistic operators to modify strings producing other strings. These operators exploit the conventions of SMILES: pairs of equal numbers indicate pairs of atoms that are joined by a chemical bond; atoms written consecutively in a word are bound; hydrogen atoms whose presence can be implied by valence rules are not written. For example, C represents methane, CCC propane, C1CC1 cyclopropane [4].

The availability of this standard provides us with an ideal object language to be used as a basis for a generative grammar.

https://doi.org/10.25088/ComplexSystems.32.4.395

Furthermore, the graphical output task can be delegated to one of the many available plotting tools accepting an input in this language.

Accordingly, we use as a set of symbols here:

$$V = \{C, 0, 1, 2, \ldots, 9\}.$$

Obviously, to keep the set $V$ of symbols finite, we must fix a limit to digit symbols; therefore, we here assume the traditional base-10 representation. No ambiguity will arise from this, because in the grammars of this work each C can be followed by at most a single number.

The rules of SMILES require that a special character # be written before a multiple-digit number to avoid ambiguity, but here we neglect this minor complication for simplicity. By the way, strings containing numbers higher than 9 describe, in the present grammars, molecules so large that most probably they are not of immediate chemical interest.

Of course, the numbers could be written instead in a binary notation; the elaboration could be described at the most "atomic" level, using machine-language or Turing machine operations; and the set of symbols would be strongly limited: $V = \{C, 0, 1\}$. This would be more elegant in the framework of formal languages theory, but we assume here, instead, that operations like "comparing" and "adding" numbers are delegated to a higher-level language, like a conventional programming language (Python, C, Fortran) endowed with integer arithmetic, to avoid many technical details.

With this proviso, let us introduce a set of Π-rules.

The "necklace" operator defined in [12] is used as the first production operator, represented here by $\delta$. It modifies its argument string X according to the following algorithm.

> Step 1: check if the substring "CC" appears in X
>
> Step 2: concatenate X to itself, that is, replace X by XX
>
> Step 3: if step 1 was successful, then: identify the highest number $m$ in X (if no number appears, then $m = 0$) and replace the first and last "CC" in XX with the expression "CpC" where p is the representation of the number $m + 1$

In the attribution of a chemical meaning to a repeating number sequence, two pairs of equal numbers are assumed to address two different couples of bound atoms. This way, number pairs are reusable. Since this convention is accepted in SMILES, no rewriting is needed.

At first glance, the operator just introduced is rather arbitrary, yet it is one of the simplest choices that satisfy these characteristics, requested by the rules of the SMILES language: equal numbers must always appear an even number of times and two equal numbers cannot appear at the beginning or end of a string.

Here is an example of an operation with $\delta$:

$\delta$(CCC1C):
Step 1: CC detected
Step 2: CCC1CCCC1C
Step 3: step 1 was successful: $m = 1$, $m + 1 = 2$,
C*C2C*1CC*C2C*1C
$\delta$(CCC1C) = CC2C1CCC2C1C

Here the pair of outermost "CC" in the doubled string has been highlighted in underlined italic.

The iterated application of the operator, shown as $\delta^{n+1} = \delta\delta^n$, produces one example of the aforementioned grammars, where the entire structure of the string determines the results.

Having defined the algorithm for applying replacement operators to a string, it is possible to perform the computation of the string resulting from the repeated operation of the operator to an initial string: the axiom $\omega$.

As can be seen from the previous example, a program for calculating $\delta^n(\omega)$ from $\omega$ would be far from long. Of course, it may take some ingenuity to actually write the needed code in a specific programming language.

There are several solutions for this implementation: one possibility is to get rid of C's and represent, for example, CC1C as a vector (0, 1, 0). The final vector is translated into SMILES; see the Appendix.

One such program, implemented in Fortran 77, has been used to produce the results of this paper and is available upon request to the author.

A second $\Pi$ operator, which can be used alone or in conjunction with the first, is $\kappa$, which generates a cyclic group. $\kappa$ takes the first C symbol in the string, followed by its accompanying numbers, and places it at the end of the string. For example:

$\kappa$(C1C2CC1C2) = C2CC1C2C1.

This operator does not change the string length, hence the number of C atoms remains the same.

The action of a power of $\kappa$ can be seen as a rotation in an abstract space of connectivity.

## 4. Selected Systems

As in [11], to illustrate the principle, we start from the simplest possible string that has a meaning in the language: in this case, the string containing only the capital letter C, which represents the methane molecule.

https://doi.org/10.25088/ComplexSystems.32.4.395

These are the first iterations of $\delta$ starting from just the atomic symbol of carbon. Note how an encoded chemical motive emerges from the initial expression:

$$\delta(\mathrm{C}) = \mathrm{CC}$$

$$\delta^2(\mathrm{C}) = \mathrm{C1CC1C}$$

$$\delta^3(\mathrm{C}) = \mathrm{C1C2C1CC1C2C1C}$$

$$\delta^4(\mathrm{C}) = \mathrm{C1C2C1C3C1C2C1CC1C2C1C3C1C2C1C}$$

Also note that the number of C atoms described by the string is doubled with each operation, and descriptions of molecules with hundreds of C atoms appear from $n = 7$.

As mentioned earlier, these strings can be translated into molecular structures using any one of the many programs available based on the SMILES language standard. The author used, as in [11] and [12], the open-access program [13] that implements the Java Script Molecular Editor (JSME) [14] and an optimization tool based on the Merck molecular force field (MMFF) [15].

The results of the first six iterations of the operator are shown in Figure 1. The molecular structure is represented in a very deformed way by the graphic tool, in an attempt to avoid the crossing of bonds in the plane. Several iterations produce a very crowded structure that might be described as a "polycyclic foam." A qualitative discussion of the structures shown highlights a few aspects: looking at the effect of successive iterations on the string, we realize that the duplication of the set of numbers is followed by the insertion of further larger numbers in intermediate positions, as schematically illustrated in Figure 1 as well. This shows that the operator duplicates and then reconnects

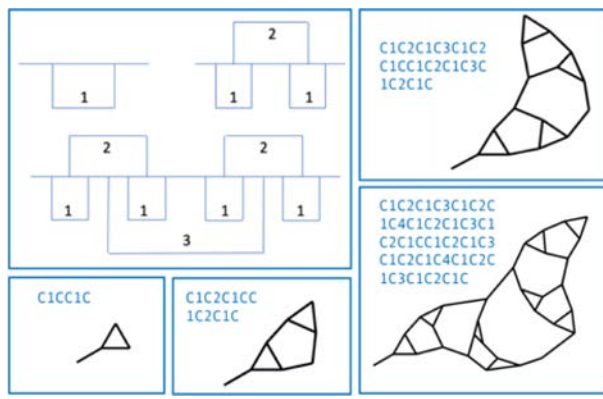


**Figure 1.** Schematics of the generation of a fractal number sequence by $\delta^n$ acting on a "C" string and strings generated by $\delta^n(\mathrm{C})$ for $n = 2$ to 5, along with corresponding molecules, plotted using the software [13].

the corresponding molecule, in a way similar to the well-known baker's map, and similarly produces a fractal structure [7]. It is important to note that the details of this numerical scheme are not universal: the scheme generated from a different axiom, such as CCC considered later, is different, although the duplication-interposition principle remains, as well as the fractal structure.

From the point of view of chemical practice, these molecular structures are quite exotic, and certainly their chemical stability is reduced by having rings with only three carbon atoms, which strain considerably the ideal geometry for $sp^3$ orbitals. It is also very difficult to imagine a possible synthesis pathway for most of them. Given the basic intent of this paper, this aspect is not essential: actually, it would be easy to add a single rule, a termination rule [11], to the algorithm to make all three-atom rings larger. This would make all the structures generated quite strain-free but would make the figures here much more difficult to read. In the huge variety of structures generated by changing $\omega$, however, some are intrinsically more stable, as seen further on.

The result of a force-field optimization calculation of the three-dimensional structure of the system $\delta^5$(C), carried out using the same program and reported in Figure 2, shows that the bonds are not hopelessly strained as it deceptively appears in the plane presentation.

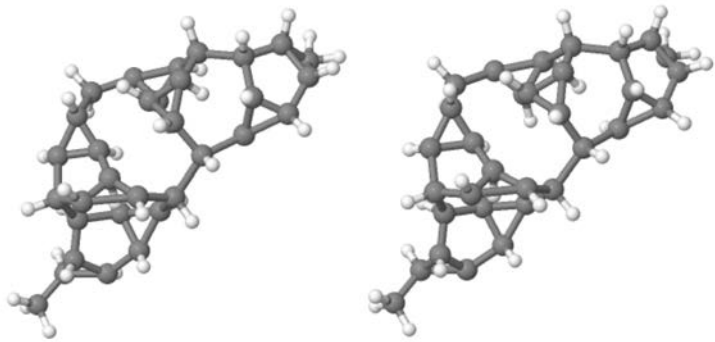

**Figure 2.** Stereogram of the structure expressed by $\delta^5$(C), rendered in three dimensions using a force field for organic molecule using the software [13]. Gray spheres are C atoms, smaller white spheres are H atoms. Local chirality is random here, left to the software as part of the rendering task.

This last figure has been generated by letting the program [13] assume the most convenient chirality for any chiral center, though it would be possible to specify it by introducing additional symbols into the string. The calculation of the string expression corresponding to successive orders remains extremely fast, but the structural optimization becomes increasingly heavy.

https://doi.org/10.25088/ComplexSystems.32.4.395

Once a string of satisfactory length has been obtained automatically, it can be processed further using the $\kappa$ operator defined in the previous section.

As can be seen from Figure 3, successive applications of $\kappa$ to $\delta^5$(C) produce nontrivial rearrangements: since several pairs of equal numbers are present in a nested arrangement, the string rotation shuffles the connectivity in the entire structure.

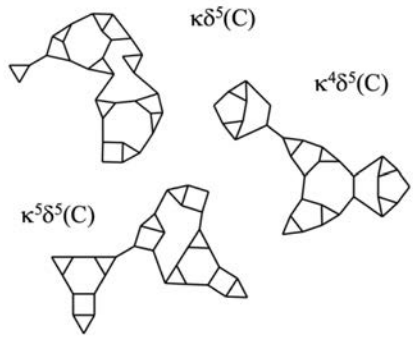


**Figure 3**. Effect of the cyclic operator $\kappa$ on the structure $\delta^5$(C). Note the radical topological rearrangements, due to the chemical convention whereby pairs of equal digits, in the order of appearance, encode bonds.

As can be seen, these structures, unlike the previous ones that reflect the fractal order of the sequence in a visible way, are decidedly more complex in the sense of being made up of numerous different parts. Between these parts are nearly flat polycycles and partially closed quasi-cages. Nevertheless, even the least predictable of such structures, being generated without omissions by a rather simple process, are characterized by low algorithmic complexity [2], a fact that could easily escape an observer unaware of such construction.

Figure 4 illustrates the reason for the radical rearrangement of the structure encoded by the new string compared to the previous one. In the figure, we imagine that the string has been linked to itself to produce a periodic circular sequence that can be read starting from any point in its entirety. Applying enough operations $\kappa$ produces the new situation on the right. The links encoded by the four occurrences of the same digits, in this case the digit 1, present in the string are all exchanged as soon as the first occurrence of the digit 1 passes to the opposite side of the reading start marker. The structure modification is global. Even more complex changes occur when the reading start marker leaves behind more than one type of digit, for example one 1 and one 2.

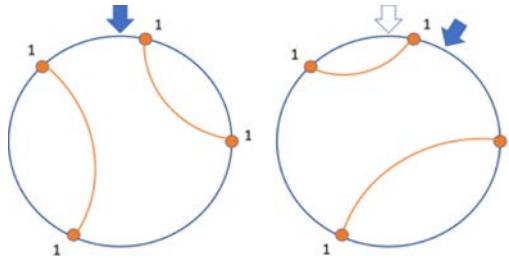


**Figure 4.** Interpretation of the effect of the $\kappa$ operator on the bonds encoded by like digits in a chemical description string based on the SMILES language. On the left, the initial situation: four occurrences of the digit 1 encode two bonds as indicated. The blue marker indicates the starting point of clockwise reading. Right: the result of $\kappa^n$, with $n$ large enough to cause the marker to leave behind the first occurrence of the digit 1.

Let us see the effect of a change of the axiom $\omega$: to stay within the context of alkane structures we can use, as $\omega$'s, strings made only of C letters, possibly but not necessarily already containing numbers. An example is $\omega$ = CCC which generates words different from those already calculated, given that the number of C's in the previous case can never be a multiple of three.

The structures that are obtained from this simple change of axiom have surprising differences compared to those already seen, also including the effect of the rotation: one observes the appearance of very large cycles and, with a simple further rotation of the string, their opening to form structures decorated by polycycles but basically linear.

Figure 5 in particular shows one of these last structures featuring numerous cycles of five atoms, partially fused. This structure, of which a preliminary force-field optimization is also presented, is a chain of different modules; one of them, its "head," is a carbon-based cage. An interesting aspect from a practical point of view is that this structure no longer has any three-atom cycle: an advantage from the point of view of likelihood, given that it is much less strained and certainly much more stable. Let us note once again that the simplest way to give this structure a name, moreover a constructive one (apart from aspects of absolute chirality), is to use its grammatical one: $\kappa^5\delta^4$(CCC). The generated SMILES line expression, equivalently non-specifying chirality, is instead 78 characters long (Figure 4). One avoiding repetitions of pairs of numbers, which is the usual practice, would be much longer. The question is left open of how varied the repertoire of structures that can be generated by ($V$, $\omega$, $\delta$, $\kappa$) is.

https://doi.org/10.25088/ComplexSystems.32.4.395

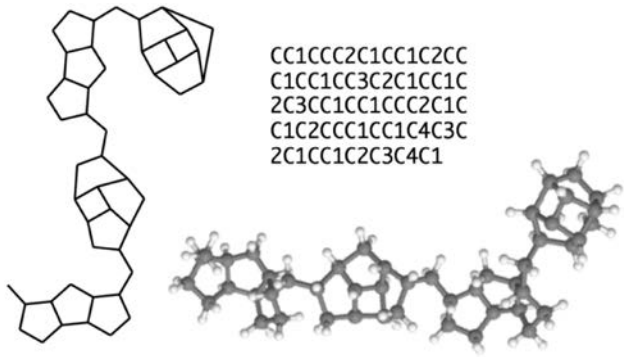


**Figure 5.** Change of the axiom. Here the structure $\kappa^5\delta^4$(CCC) is shown, which lacks 3-cycles and features several 5-cycles, with a considerable gain in stability due to lower bond strain. Its "head" (encoded into the tail of the string) is a carbon cage. All these characteristics, which emerge a posteriori, are encoded by the expression $\kappa^5\delta^4$(CCC). Structures satisfying different requirements could be detected in the space of generated structures by a suitable program.

## 5. Conclusion

The main message of this paper is that simple programs can generate linear descriptions of remarkably complex molecular structures.

This message has practical consequences: consider the advantage of specifying a structure as large, disordered, but locally organized as desired, to be used as a model substrate in *ab initio* calculations, by means of a concise expression. Consider also the possibility of varying this structure within a certain motif, in a radical but reproducible way, by modifying the parameters of the same description.

This may also provide an interpretive tool for complex structures: the search for simple algorithms that generate at least part of it.

This result may support recent criticism (e.g., [16]) of the assembly theory (AT) cited in the introduction. Let us preliminarily observe that even if in this paper we have only considered hydrocarbon structures, and of a very restricted class, a previous work by the author has shown that these operators can build much more varied molecules [12] and can be further developed to describe complex biomolecules. The radical upheaval of the topology of a molecule encoded by a cyclic string, like in Figure 4, by the application of the operator $\kappa^n$ could have allowed primordial organisms to rapidly adapt to new situations by producing radically new enzymes and building blocks. This versatility of cyclic chemical encoding may have been a reason for the emergence of cyclic RNAs in primordial life forms [17]. In this

perspective, the result of the operation of $\kappa^n$ is the complete dismantling and rebuilding of its smaller constitutive blocks in a way difficult to capture by the formalism of AT based on the assembly index alone.

This paper is only a first exposition of the idea. Other operators can be formulated. We only show here two possible axioms $\omega$ out of the huge variety of those possible. We limited our elaborations to a low number of iterations of $\delta$, with the idea of showing visibly complex but still readable structures. In fact, there is no limitation, provided effective string-to-structure translation tools and enough computing power are available. Finally, no attempt has been made to analyze in detail the structure of the numerical sequences that can be produced, which is likely related to number theory. The ideas presented in this paper also may impact on the usual application of L-systems in theoretical biology.

### Appendix: A Program Implementing the Grammar ($V$, $\omega$, $\delta$, $\kappa$)

As mentioned in the main text, one way, not the only one, to implement the grammar ($V$, $\omega$, $\delta$, $\kappa$) is to use number sequences, obtained by removing all C's and writing a C not followed by a number as 0. A number sequence can be represented in a language like Fortran 77 by a vector $N$ of integers, of length $L$, embedded into a predefined large vector. If any 0 not in the last position is located, then a CC is detected (a final 0 does not represent CC but C). To double the sequence, each $N(i)$ is copied into $N(i + L)$ to obtain a vector of $2L$ elements. The first 0 found by searching from the first element and the first 0 found by searching backward from $N(2L - 1)$ are located. These two 0s are both replaced with the number $m + 1$. At the end of a calculation, the sequence is translated into a SMILES string by adding C's and removing 0s. String rotation $\kappa$ is directly implemented. The actual Fortran 77 program is a few lines long and calculates any word generated by ($V$, $\omega$, $\delta$, $\kappa$).

## References

[1] G. P. Moss, "Extension and Revision of the von Baeyer System for Naming Polycyclic Compounds (Including Bicyclic Compounds)," *Pure and Applied Chemistry*, **71**(3), 1999 pp. 513–529. doi:10.1351/pac199971030513.

[2] G. J. Chaitin, "Randomness and Mathematical Proof," *Scientific American*, **232**(5), 1975 pp. 47–53. doi:10.1038/scientificamerican0575-47.

[3] M. Li and P. M. B. Vitanyi, "Kolmogorov Complexity and Its Applications," *Algorithms and Complexity* (J. van Leeuwen, ed.), Amsterdam: Elsevier Science Publishers, 1990 pp. 187–254. doi:10.1016/B978-0-444-88071-0.50009-6.

[4] D. Weininger, "SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules," *Journal of Chemical Information and Computer Sciences*, **28**(1), 1988 pp. 31–36. doi:10.1021/ci00057a005.

[5] S. M. Marshall, C. Mathis, E. Carrick, G. Keenan, G. J. T. Cooper, H. Graham, M. Craven et al., "Identifying Molecules as Biosignatures with Assembly Theory and Mass Spectrometry," *Nature Communications*, **12**(1), 2021 3033. doi:10.1038/s41467-021-23258-x.

[6] G. E. Révész, *Introduction to Formal Languages*, North Chelmsford: Courier Corporation, 1991.

[7] M. R. Schroeder, *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*, New York : W. H. Freeman, 1991.

[8] A. Lindenmayer, "Developmental Algorithms for Multicellular Organisms: A Survey of L-systems," *Journal of Theoretical Biology*, **54**(1), 1975 pp. 3–22. doi:10.1016/S0022-5193(75)80051-8.

[9] G. S. Hornby and J. B. Pollack, "Evolving L-systems to Generate Virtual Creatures," *Computers and Graphics*, **25**(6), 2001 pp. 1041–1048. doi:10.1016/S0097-8493(01)00157-1.

[10] P. Prusinkiewicz, M. Cieslak, P. Ferraro and J. Hanan, "Modeling Plant Development with L-systems," *Mathematical Modelling in Plant Biology* (R. Morris, ed.), Cham: Springer, 2018 pp. 139–169. doi:0.1007/978-3-319-99070-5_8.

[11] S. Longo, "Generative Grammars for Branched Molecular Structures," *Chemical Physics Letters*, **809**, 2022 140151. doi:10.1016/j.cplett.2022.140151.

[12] S. Longo, "Molecular Blueprinting by Word Processing," *Symmetry*, **15**(2), 2023 357. doi:10.3390/sym15020357

[13] P. Ertl and B. Bienfait. "DIY-molecules: Build Your Own Molecule." (Jan 24, 2024) biomodel.uah.es/en/DIY/JSME/draw.en.htm.

[14] B. Bienfait and P. Ertl "JSME: A Free Molecule Editor in JavaScript," *Journal of Cheminformatics*, **5**(1), 2013 24. doi:10.1186/1758-2946-5-24.

[15] T. A. Halgren, "Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94," *Journal of Computational Chemistry*, **17**(5–6), 1996 pp. 490–519. doi:10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P.

[16] A. Uthamacumaran, F. S. Abrahão, N. A. Kiani and H. Zenil, "On the Salient Limitations of the Methods of Assembly Theory and Their Classification of Molecular Biosignatures." arxiv.org/abs/2210.00901.

[17] X. Meng, X. Li, P. Zhang, J. Wang, Y. Zhou and M. Chen, "Circular RNA: An Emerging Key Player in RNA World," *Briefings in Bioinformatics*, **18**(4), 2017 pp. 547–557. doi:10.1093/bib/bbw045.