

Modeling Plant Growth with String Rewriting Algorithms

Florian Lorétan, Earlham College

Fall 2006

1 Introduction

The plant kingdom offers us with a great diversity of shapes, colors and practical functions, but all plants are at their initial stage very similar: a seed that germinates and that will turn into a plant if the conditions are favorable. An important characteristic of plants is that they are sessile (i.e. they do not move). Whereas most other living organisms can adapt to changes in their environment by moving, plants need to adapt to their environment in the way they grow. It is this adaptation that results in the wide diversity of plant species.

Understanding growth processes is crucial in order to understand plant development. This understanding needs to be qualitative but also quantitative. Mathematical models provide us with the necessary tools to express our assumptions in a formal way.

Beyond their simple appearance, plants are complex structures whose functioning is often very obscure even today. Mathematical models, in order to have a practical value, need to abstract from the unnecessary details, but they still need to be detailed enough to provide us with solutions to our problems that match reality. The use of computers to effectuate the computation required for using a model has pushed back the limit of what amount of detail can be included while keeping the computations feasible.

In this paper, I will give an overview of some of the mathematical models used in biology and more specifically in botany in order to compare the advantages and disadvantages of string rewriting. I will then give an explanation of the principles of string rewriting and give an account of my modeling

project using such a model to study the growth of the plant *Floerkea Proserpinacoides*.

2 Motivation

2.1 Mathematical Modeling

In order to give a broad overview of mathematical models used in biology, we will show different techniques that can be used to model population growth.

2.2 Aggregate Models

Let us first consider the population to be an undifferentiated mass. In this case we only care about the size of the total population. Our assumption about the dynamics of that population is that its growth will be proportional to its size, in other words, the bigger the population the faster it will grow. We can write this constraint as a differential equation of the form:

$$\frac{dy}{dt} = \alpha \cdot y$$

where α is a constant expressing the rate of growth. By solving this equation analytically we get the size of the population as a function of time:

$$y(t) = c \cdot e^{rt}$$

Here c is a parameter indicating the original population at time $t = 0$, and r reflects the growth of the population. If r is positive then the population will grow exponentially without bounds, a situation that cannot happen in real life and obviously shows a weakness of our simplistic model.

If we look at the causes of growth more in detail we can attribute it to reproduction, an event occurring at discrete intervals. We can then turn our differential equation into a difference equation. The constant α is now replaced with another constant β that indicates the proportion of the population that will reproduce during one time interval:

$$x_{n+1} = x_n(1 + \beta)$$

With this formula, we can easily compute a sequence of generations given a starting population. The model remains however aggregate, as there is no

differentiation between the behavior of individuals, they all follow the same probability.

2.3 Individual Models

If our goal is to mimic the behavior of each individual parts independently, we need to make that behavior dependant on local conditions and not on the whole population. Let us refine our population growth example to the case of a bacteria colony which cannot grow without having a certain amount of space in order to avoid overcrowding. If we represent each bacteria as a square cell in a grid on a two-dimensional plane, we can define the growth restriction by giving as a rule that the colony will only expand to an adjacent square on the grid only if this one is adjacent to one and only one occupied square. Figure 1 shows the first few steps of the growth of a bacteria colony starting with only one individual in an empty plane.

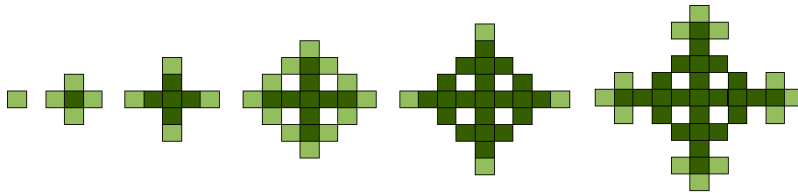


Figure 1: An example of cellular automata.

The rule is fairly simple and easy to compute, but it has to be repeated many times, once for every square in the plane. So we have traded analytical complexity for computational complexity, but we have also gained a more detailed model.

Computers are a tremendous help for such computations that are simple and repetitive, and there are many ways of implementing discrete models. One of them is to use a string rewriting algorithm. This method was used by Aristid Lindenmayer in 1968 to model the growth of simple organisms. The procedure was later successfully applied to higher plants and is now known as L-System [3].

2.4 Validation

At the base of the scientific method is the idea of making hypotheses based on observations and prove their validity by doing empirical experiments. The



Figure 2: a picture of a shell taken with a camera (left) and a picture rendered from a mathematical model (right).

process of creating a mathematical model for a natural process follows the same procedure. If a model is based on empirical observations and measurements, its validity cannot be guaranteed until it has been shown that the behavior predicted by the model matches with an observed independent data set. Depending on what aspect of a process we want to model and what the purpose of the model is, the form of the independent data set will vary; if the goal is to model the statistical growth of a population, we will have to compare the numbers generated by our theoretical model to the population count of a sample of such a population, and if we care about the way some organism grows (without trying to quantify it) we will need to compare the generated structure with the structure of existing organisms. Figure 2 shows an example of comparison at one point in parameter-space.

Note that comparing a model to an already existing valid model for the same process can be sufficient to prove the validity of the new model, without having to resort to a possibly more complicated comparison with the original data set that was used to prove the validity of the original model.

Theoretical models become interesting when we can establish a link between their mathematical and observed physical parameters, allowing one to study a cause-effect relationship. However, whenever we include such parameters in a model, we also need to validate it according to all relevant values for that parameter. So for example, in the model of the shell given above, if we assume that the pattern is influenced by the quantity of dissolved iron in the water, then we will need to compare the results of our model with

samples from the whole range of values that the iron solution can take.

One would be tempted to make models that involve many parameters with the intention of making it as realistic as possible. However, every time an additional parameter is included in the model, we need to test the model for relevant values of this parameter in combination to all the other sets of values from the other parameters. Each time we add a parameter to a model, we add one dimension for the parameter space for which we need to compare our model to an independent data set. Validating models with many parameters quickly becomes difficult, as either the computational complexity or the complexity of gathering the data necessary for the comparisons sets a bound on what is feasible. We might be able to generate models of arbitrary complexity, but they have no scientific value until they are validated.

3 Project

This project consisted of investigating the possibilities for implementing mathematical models for plant growth using string rewriting algorithms in a computationally efficient way. We will now discuss the different aspects of this modeling experiment.

3.1 *Floerkea Proserpinacoides*

Due to the extreme diversity of the plant kingdom, there is no plant specie that is a good representative of plants in general or even of a group of plants. According to B. Smith (personal interview, October 26, 2006) choosing an arbitrary plant for a modeling experiment is therefore a reasonable decision.

However, *Floerkea Proserpinacoides* does have some advantages that make it an interesting plant to model. Its structure is relatively simple and it has little direct interaction with animals or other plant species. It does have an interesting nondeterministic behavior which we will discuss later in this section.

All the data used for generating the model and for validating it come from Smith's series of three articles [4, 5, 6].

This small plant grows on the floor of forests in the north of the United States early in the year, when bigger trees have not yet developed leave that keep the light out of the lower layers of the forest.

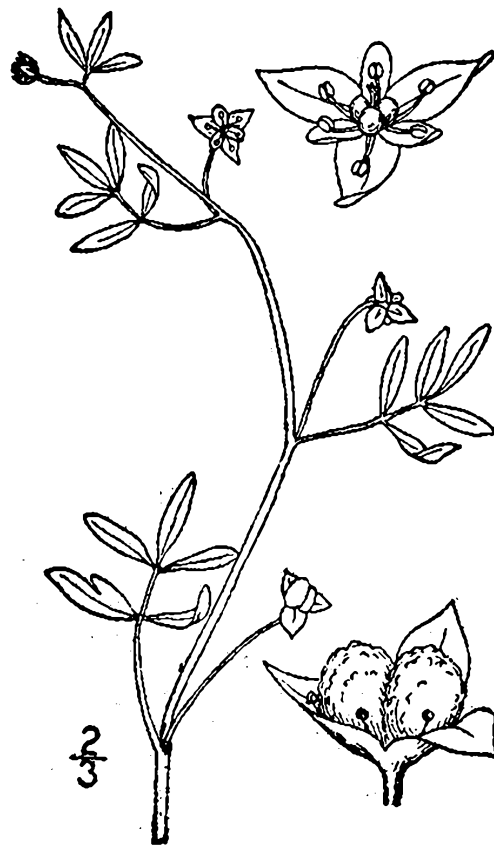


Figure 3: *Floerkea Proserpinacoides*, commonly known as *false mermaidweed*

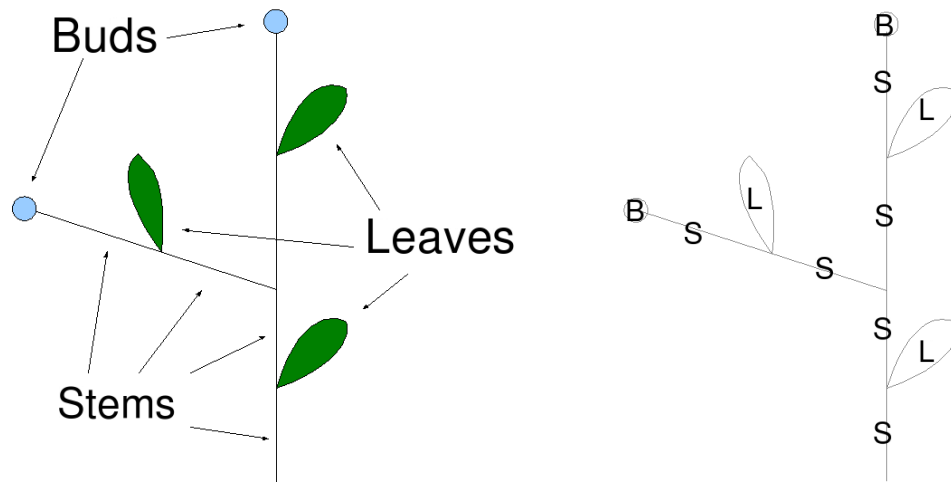


Figure 4: Identifying each part of the plant and associating a symbol to it.

3.2 String Rewriting

String rewriting algorithms are based on our ability to:

1. Describe a certain structure by a sequence of symbols.
2. Express modifications to that structure by giving replacement rules for the symbols used in the descriptive string.

For more clarity, let us look at a simple example. Consider the hypothetical plant shown in Figure 4. We identify each part of a plant and associate a certain symbol with it. We now want to put these symbols in a sequence while preserving the structure. To do so, we start by the bottom of the plant and start listing the symbols by following the structure of the plant. The opening bracket symbol '[' is used to indicate the start of a new branch. Once the branch is described, we close the bracket with ']' to indicate that the description now continues from where we branched off. Once we have parsed the whole structure of the plant we get the following string:

$$S L S [S L S B] S L S B$$

Now that we have a way to establish a correspondence between plant parts and symbols in a string, we can define growth rules by writing down the corresponding replacement rules for symbols. For example, a rule might be that buds B turn into flowers F . We can write this in the following way:

$$B \mapsto F$$

Using this rule to process our string we get the description of a plant in which all buds have turned into flowers:

$$S L S [S L S F] S L S F$$

The whole string rewriting algorithm can be decomposed into four different steps shown in figure 5, all of which are independant of the structure we are modeling.

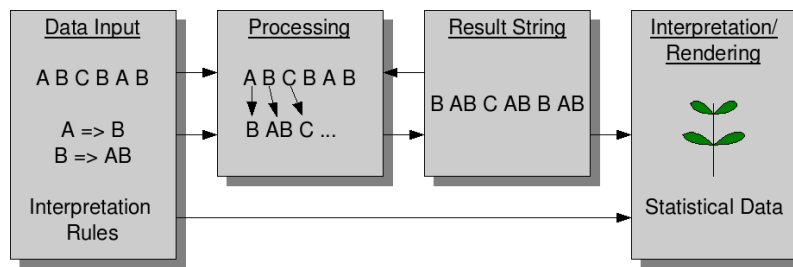


Figure 5: The different steps involved in a string rewriting application.

Data Input: Here we define the *axiom*, the original string that we will feed to the string processor before the first iteration. We also define the rules by matching symbols to sequences of symbols. Finally, we define interpretation rules establishing the relationship between the symbols and what they represent, so that we have a way to transform a string of symbols back into an understandable form.

String Processing: The string processor takes an input string and compares every symbol of it to the left-hand side of the replacement rules that have been defined. If there is a match then the symbol is replaced by the right-hand side of that rule. In case no rule matches a symbol, that symbol is left in string.

Result String: The result string is the output of the string processor after all rules have been applied. This result string can be fed back into the string processor in order to generate more than one iteration. It can also be transferred to the renderer/interpreter.

Rendering/Interpretation: The result string is an abstract representation of some structure which is not readily understandable by humans. The transformation is done by matching symbols on the string to instructions given to some rendering engine that transforms them into images or any kind of data.

Sometime only the end result of the string processor after a certain amount of iterations is meaningful, but sometime every iteration gives a meaningful result. In that case, rendering the output of the string processor before feeding it back to it shows the evolution of our model through every iteration.

3.3 Nondeterminism

A nondeterministic process is one whose result is not predictable. Such processes are common in plant growth when the exact causes of a certain behavior are not known, but the possible different outcomes is known as well as the proportions in which they occur.

Floerkea shows a particular example of such a nondeterministic behavior [7]. The first two nodes create branches or nothing, and the nodes with numbers higher than 4 create a flower or nothing, but the third node develops a branch in some cases and a flower in other cases.

This little structural difference happens to be critical for the survival of the specie, as it will not only effect the further development of the plant, but also the amount of seeds produced. In fact, the creation of a branch will generate more seeds if given the proper time to develop, but developing a leaf and focusing on the vertical development of the plant is a safer behavior. In his study, Smith found that the development of a branch or a leaf was related to the density of the forest surrounding the plant.

Using string rewriting, this behavior can be implemented by assigning two different rules to the same symbol, each rule having a certain probability associated with it in order to indicate with in what proportions the rules are applied. The nondeterministic rule for the third node of *Floerkea* can be described in the following way:

$$\begin{aligned} N &\mapsto F \text{ (60\%)} \\ N &\mapsto B \text{ (40\%)} \end{aligned}$$

Notice that in this case the sum of the proportion of the possible outcomes has to be 100%. We can loosen that restriction by using weights instead of proportions, in which case the proportion of the time when a certain rule is applied is relative to the weight of the other rules.

If we replace fixed proportions with functions of external parameters such as the amount of light reaching the ground, we can extend our model and make it much more flexible.

3.4 Experiment Design

The experiment that will be simulated is the growth of a Floerkea in standard conditions. There is some data available regarding the influence of density and light availability, but these parameters will not be considered at this point.

Table 1 gives a basic listing of the replacement rules used for the Floerkea model. The rules actually used for the computation include more details such as the limitation of branches to the first three nodes, but they are not displayed here for clarity purposes.

Predecessor		Successor
Seed	\mapsto	Root + Stem + Apex
Apex	\mapsto	Leaf + Node + Stem + Apex
		Bud (70%)
Node	\mapsto	or
		Inactive Node (30%)
		Flower (60%)
Bud	\mapsto	or
		Branch (40%)
Flower	\mapsto	Seeds

Table 1: A basic set of replacement rules for the growth model of Floerkea.

This model gives rules for the growth of a plant but not for its decay. The reason for this is that Floerkea has a fixed growing season and that the death of the plant happens when the leaves of the trees make the canopy too dense for a sufficient amount of light to reach the forest ground. It is therefore a reasonable simplification to assume that the plant will suddenly stop growing at that moment and that no further changes to the plant structure will be made until its death.

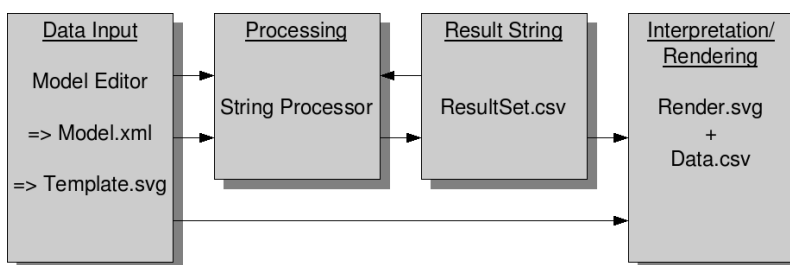


Figure 6: Software implementation.

3.5 Software Implementation

There are already a few existing software packages available that deal with string rewriting and L-systems, some of them such as vlab (for Unix/Linux) or L-Studio (for Windows) being specialized for plant modeling. However, instead of relying on the use of one of these programs I decided to create my own implementation of a recursive string rewriter, the benefits being a direct control on the behavior of my program and a deeper understanding of the processes involved as well as the freedom of open source software. Most of the existing applications also focus on graphical uses of plant models, whereas the focus of this project is on the scientific validity of the models.

The implementation used for this project is shown in figure 6 and reflects the general structure outlined in figure 5.

Data Input: The model editor provides a graphical user interface for defining the axiom and the rules, as well as the number of iterations to be computed, and saves this information in an XML (Extensible Markup Language) file. A template SVG (Scalable Vector Graphics) file is used to define the graphical interpretation rules of every symbol to be used in the rendering process.

String Processing: This is where the computation actually happen. The methods and data structures used here are discussed in the next section.

Result String: Since we have some nondeterministic rules, we want the possibility to keep track of the many possible strings that can be generated. The CSV (Comma Separated Values) file is therefore a result set containing possible result strings, with a proportion associated to each one of them.

Rendering/Interpretation: This part of the process is split into two sub-parts. First the template file defined in the data input step is used to generate a graphical representation of the plant structure which is used for visual verification. Another module parses through result strings and extracts statistical data such as the average length over a population or the occurrence of various symbols, indicating the occurrence of the corresponding plant parts.

3.6 Method

In order to study the properties of the whole population, we need to keep track of all the possible plants that can be generated by our model. That means that every time a nondeterministic rule can be applied we need to branch off and consider every possible results string separately for which we need to keep track of its proportionality to the whole population. This process results in a directed acyclic graph similar to the one shown in figure 7. Notice that this is not a tree structure because the same results string can be generated using different paths, causing branches to merge. The proportion of a result string to the total population is the sum of the proportions of every path that leads to it. The sum of the proportions of all result strings generated after any amount of iterations is always equal 100%.

The issue with this method is that it has an exponential complexity both in terms of time and space. Most results strings are usually very similar to one another with just the end being different. Reducing the redundancy of the information can save memory space and also computation time because the same information does not need to be processed more than once.

A *trie* data structure (the name is derived from *retrieval* due to the common use of that data structure in lookup tables). A trie stores strings in the nodes of a tree and branches off whenever two or more different strings exist. Every leaf of the tree then represents one result string with an associated proportion.

In this case switching from a simple array to a trie data structure reduced the amount of information needing to be stored down to 10% of the original. However, due to the exponential complexity of our algorithm, this improvement only made one more iteration possible, with a population containing 76,440 different variations.

In order to be able to gather information about the same model using more iterations, a different approach to the complexity issue is required.

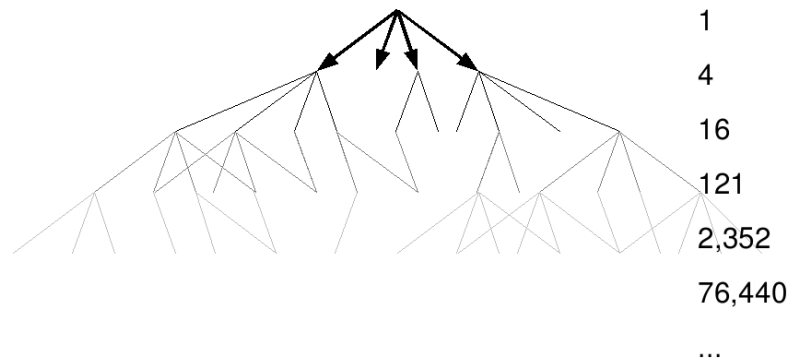


Figure 7: Breadth-first approach: explore all possible versions of the plant using breadth-first generation.

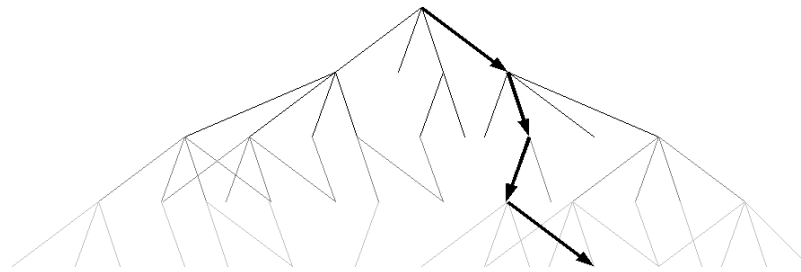


Figure 8: Depth-first approach: sample the population by generating individuals separately.

Instead of generating the whole population, it is possible to generate one individual at a time as is showed in figure 8. To do so, every time we are presented with a nondeterministic choice we make a random but weighted decision on which rule to apply. In this case the proportions are now turned into probabilities.

Instead of generating the whole population, we are going to take a sample of it in the same way that botanists study a sample of the real population (in reality, the “whole population” does not exist). This is done by running a fixed number of depth-first string generation in which only one result string is returned. By basing the decisions on a truly random source we know that any two samples are unlikely to have the same result string and that our sample is going to be homogeneous with respect to the population.

An important difference between the breadth-first and depth-first ap-

proaches is that the first one models the proportions of different kinds of individuals in the population, whereas the latter models an actual group of individuals. That means that when comparing the data from the two approaches we need to weight the data from the breadth-first approach according to the proportions associated with each result string. We don't need to do anything for the results gathered from the depth-first approach since the weights have already been applied when choosing the replacement rules during the processing.

3.7 Results

Rendering the result strings graphically gives us a visual verification which shows us that the expected behavior of the model is happening. Notice for example on figure 9 that branches appear on the first three nodes but never at higher nodes.

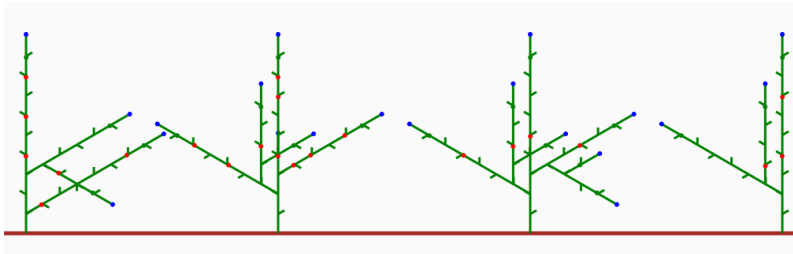


Figure 9: Schematic representations of different possible plant structures.

We also want to make sure that the breadth-first and depth-first computation give similar results. Table 2 shows a comparison of the two sets of data, and we see that with a sampling size only equal to 5% of the set of all possibilities we can already have a precision of one to two decimal places.

4 Conclusion

Modeling plant growth using string rewriting is a very powerful process that has been showed to have many different applications ranging from scientific models used in biology to graphical models used in realistic 3D image generation. This method is also flexible and easily extended. Improvements such

Symbol	Description	Whole Population	Random Sample
A	Apices	2.50	2.48
B	Buds	2.25	2.25
F	Flowers	1.72	1.71
I	Internodes	12.27	12.18
L	Leaves	9.44	9.42
N	Nodes	3.40	3.47
R	Root	1.00	1.00
A2	Secondary Apices	0.32	0.28
B2	Secondary Buds	0.25	0.23
[and]	Branches	9.45	9.42
Total String Length		53.27	52.04

Table 2: Comparison between average statistics (in number of symbols) taken on the whole population (2353 possibilities) and a random sample (100 individuals), each after 6 iterations.

as context sensitive rewriting expand the possibilities of the various processes that we can simulate.

String rewriting has not a scope limited to modeling plants, but any kind of structure that could be described by a string of symbols. The method has indeed been used to generate music as well as architectural drawings.

If some of these applications have little need for accuracy, scientific models do need accurate data in order to give us a high level of confidence in their validity. The data used for this project was not originally meant to be used in a string rewriting model and adapting it to that purpose has been challenging. I hope to have the opportunity to do more work modeling plant growth and to cooperate with botanists in order to gather data that would allow us to refine and further calibrate the model presented in this paper.

Contents

1	Introduction	1
2	Motivation	2
2.1	Mathematical Modeling	2
2.2	Aggregate Models	2
2.3	Individual Models	3
2.4	Validation	3
3	Project	5
3.1	Floerkea Proserpinacoides	5
3.2	String Rewriting	7
3.3	Nondeterminism	9
3.4	Experiment Design	10
3.5	Software Implementation	11
3.6	Method	12
3.7	Results	14
4	Conclusion	14

List of Figures

1	Cellular automata	3
2	Visual verification	4
3	Floerkea Proserpinacoides	6
4	Example Plant	7
5	Data flow of string rewriting	8
6	Software implementation.	11
7	Breadth-first generation	13
8	Depth-first generation	13
9	Schematic representation	14

List of Tables

1	Replacement rules	10
2	Data comparison	15

References

- [1] JOHN E. HOPCROFT, RAJEEV MOTWANI, J. D. U. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.
- [2] MURRAY, J. *Mathematical Biology*. Springer-Verlag, New York, 2002.
- [3] PRZEMYSŁAW PRUSINKIEWICZ, A. L. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1996.
- [4] SMITH, B. H. Demography of floerkea proserpinacoides, a forest-floor annual: 1. density-dependent growth and mortality. *The Journal of Ecology* 71, 2 (1983), 391–404.
- [5] SMITH, B. H. Demography of floerkea proserpinacoides, a forest-floor annual: 2. density-dependent reproduction. *The Journal of Ecology* 71, 2 (1983), 405–412.
- [6] SMITH, B. H. Demography of floerkea proserpinacoides, a forest-floor annual: 3. dynamics of seed and seedling population. *The Journal of Ecology* 71, 2 (1983), 413–425.
- [7] SMITH, B. H. The optimal design of a herbaceous body. *The American Naturalist* 123, 2 (Feb. 1984), 197–211.