Designing with L-Systems, Part 1: String Rewriting Systems

Strings

A *string* is a sequence of characters, such as a word, a phrase, or a telephone number. The characters may be letters, digits, punctuation marks, dollar signs, and so forth. Uppercase letters are used in examples to make them stand out, as in ABCBA.

The characters in strings may or may not have meanings associated with them. For the time being, they will just be abstract symbols. Strings may of course, contain patterns; in fact, this is the major interest here. For example, the string ABCBA is a palindrome, reading the same way forward and backward. Another example is DEDEDEDE, which consists of four repetitions of DE.

A string within a string is called a *substring*. For example, DE, ED, and DED are among the many substrings in DEDEDEDE.

The length of a string is the number of characters in it. For example, the length of ABCBA is 5. The *empty string*, consisting of no characters, has length 0. The empty string is denoted by Θ.

A single character is a one-character string.

Concatenation consists of appending one string to another. For example, the concatenation of ABCBA and DE produces ABCBADE.

Formal Languages

A formal grammar is a system that produces a *language*.

A language is a set of strings. For example, {DE, DEDE, DEDEDE, …} is the language of strings that are repetitions of DE.

The set of symbols used in a language is called its *alphabet*.

Symbol is an abstract concept. In the discussion that follows, symbols are represented by characters.

String Rewriting Systems

A *string rewriting system* consists of an initial string, called the *seed*, and a set of rules for specifying how the symbols in a string are rewritten as (replaced by) strings.

Here is an example string rewriting system:

seed:	ABCD
rules:	A → BD
	B → B
	C → ACA
	D → Θ

The character → means "is replaced by".

The first rule specifies that A is replaced by BD. The second rule specifies that B is replaced by B; that is, it is unchanged. The third rule specifies that C is replaced by ACA, while the fourth rule specified that D is to be replaced by the empty string; that is, deleted.

Given the seed ABCD, the first rule would change it to BDBCD, the second rule would leave it unchanged, the third rule would change it to ABACAD, while the fourth rule would change it to ABC.

As a matter of convention, if there is no rule for a character, that character is left unchanged. Thus, the rule B → B could be omitted.

When a string rewriting system is used, the rules are applied to the seed to produce a new string, the rules are again applied to this string to produce another, and so on, forming a sequence of *generations*, with the seed being considered generation 0. The sequence of generations constitute the language for the rewriting system. *Note:* In sets, duplicates don't count.

The languages of most rewriting systems of interest are infinite, with each generation producing a new, usually longer string.

Different kinds of rewriting systems apply rules in different ways. For example, some rewriting systems pick a rule at random to produce the next generation. For the example rewriting system given above, the generation might go like this:

string	*rule*	*generation*
ABCD		0
BDBCD	1	1
BDBCD	2	2
BBC	4	3
BBACA	3	4

BBBDACABD	1	5
BBBACAB	4	6
BBBAACAAB	3	7
…	…	…

Note that in this generation, there is a duplicate string and some strings are shorter than their predecessors.

L-Systems

Lindenmayer systems, or L-Systems for short, are named after their inventor, Aristid Lindenmayer. See the side bar. The distinguishing characteristic of L-Systems is that all rules are applied in parallel for each generation.

For the example in the last section, the generation goes like this:

string	*generation*
ABCD	0
BDBACA	1
BBBDACABD	2
BBBBDACABDB	3
BBBBBDACABDBB	4
BBBBBBDACABDBBB	5
BBBBBBBDACABDBBBB	6
…	…

As you'd expect, Bs accumulate and each generation is longer than the previous one.

Despite their apparent simplicity, L-Systems are very powerful. Their languages may contain intricate patterns with infinitely varying subtlety. Among other things, they can describe plant development (their original use), fractals, and complex geometric designs. See the Appendix.

The power of L-Systems comes from parallel rewriting and repeated application (iteration) of the rules. These properties are of fundamental importance and apply to entirely different mechanisms, such as cellular automata [1] and to many processes in the physical world.

Aristid Lindenmayer

1925 - 1985

Aristid Lindenmayer was a Hungarian biologist. His invention of the string rewriting system named after him "grew out of an attempt to describe the development of multicellular organisms in a manner which takes genetic, cytological and physiological observations into account in addition to purely morphological ones" [2].

It was a surprise to him when his system began to be used in many other disciplines and in many ways. In fact, he first rejected as a gimmick the graphical interpretation of L-System strings to draw plants, something he never envisioned when he started to manipulate strings of characters. Drawing ultimately proved to be the most influential use of L-Systems.

There are L-Systems of many different kinds and degrees of complexity, including ones that can be used to represent three-dimensional plants in color and geometrical forms [3-5]. Active research in the area continues to this day.

Example L-Systems

Example 1: The Morse-Thue sequence [6] is produced by a very simple L-System:

seed:	A
rules:	A → AB
	B → BA

The generation goes like this

```
A
AB
ABBA
ABBABAAB
ABBABAABBAABABBA
      …
```

Note that the lengths of the strings double with each generation.

Example 2: The Fibonacci string sequence, analogous to the Fibonacci number sequence [7], is produced by this L-System:

seed:	A
rules:	A → B
	B → AB

Generation goes like this:

```
A
B
AB
BAB
ABBAB
BABABBAB
ABBABBABABBAB
BABABBABABBABBABABBAB
            …
```

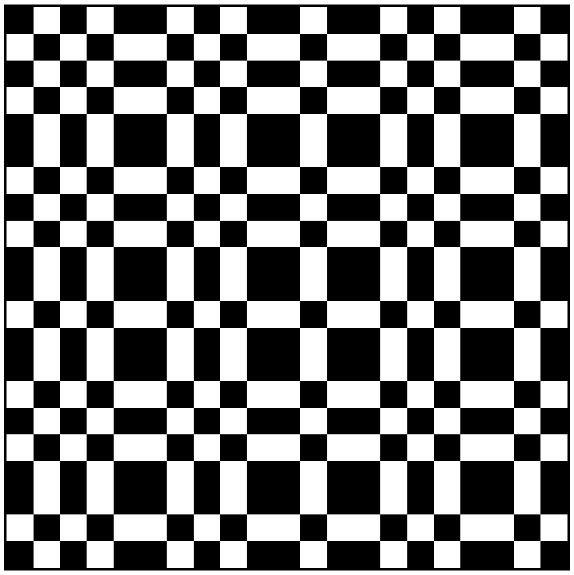
Note that the lengths of the generations give the Fibonacci numbers: 1, 1, 2, 3, 5, 8, 13, 21, … .

Interpreting L-System Languages

The strings produced by L-Systems such as those given above have evident patterns but the characters themselves have no meaning. If A and B in the Morse-Thue example are interpreted as 0 and 1, respectively, the results is the usual interpretation of the Morse-Thue sequence as a sequence of binary digits.

Many other kinds of interpretation are possible. One of the most striking methods interprets characters as drawing commands. The examples in the Appendix were produced in this way. The next article will describe the drawing interpretation.

A very natural interpretation of strings like the Fibonacci strings is as profile sequences. Here is a profile pattern for the seventh-generation Fibonacci string:



There are many other possibilities, both in the crafting of L-Systems for particular purposes and in interpreting them. These will be topics of future articles.

References

1. *Drawdown Automata, Part 1: Basic Concepts*, Ralph E. Griswold, 2002:
http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_dda1.pdf

2. "Developmental Systems and Languages in their Biological Context", Aristid Lindenmayer in *Developmental Systems and Languages*, G. T. Herman and G. Rozenberg, eds., North-Holland/American Elsevier, 1975.

3. *The Book of L*, G. Rozenberg and A. Salomaa, eds., Springer-Verlag, 1986.

4. *Lindenmayer Systems, Fractals, and Plants*, Przemyslaw Prusinkiewicz and James Hanan, Spring-Verlag, 1989.

5. *The Algorithmic Beauty of Plants*, Przemyslaw Prusinkiewicz and Aristid Lindenmayer. Springer, New York, 1990

6. *The Morse-Thue Sequence*, Ralph E. Griswold, 2004:
http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_mt.pdf

7. *Drafting with Sequences*, Ralph E. Griswold, 2004:
http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_seqd.pdf

Ralph E. Griswold

Department of Computer Science
The University of Arizona
Tucson, Arizona

Appendix — L-System Graphics