Designing with L-Systems, Part 4: Articulated L-Systems

Variables and Constants

L-Systems [1] have no concept of characters that play different roles. Most rewriting systems distinguish between variables and constants. Variables have associated replacement strings; constants do not and just stand for themselves.

In an L-system, if there is no rule for a character, it is replaced by itself. In practical terms, it's a constant.

Articulated L-Systems distinguish between variables and constants. The mechanism for rewriting and producing successive generation remains the same. The only difference is in the classification of characters. In articulated L-Systems, characters are divided into two classes as described above: constants and variables.

This L-System from the article on using L-Systems to produce graphic images [2] illustrates the difference:

seed: X
rules: X → F–[[X]+X]+F[+FX]–X
F → FF

Here X and F are variables, while –, +, [, and] are constants.

To help distinguish variables from constants in examples that follow, uppercase letters are used for variables and all other characters are constants. This typographic distinction is just a matter of convenience; there is nothing fundamental about it.

Constants are just what their name implies. They are not replaced in rewriting (which is a more useful idea than the one that they are replaced by themselves).

Defined and Undefined Variables

Variables in turn are divided into two classes: defined and undefined. A defined variable is one for which there is a rewriting rule. An undefined variable is one that appears in an L-System but for which there is no rewriting rule. During rewriting, undefined variables are treated like constants, but they play a different conceptual role.

Previous examples of L-Systems have had no undefined variables. An undefined variable can serve two purposes. One is as a placeholder for a long constant string. An example is

seed: S
rules: S → STS

Here, T is an undefined variable. Successive generations are

S
STS
STSTSTS
STSTSTSTSTSTSTS
…

If T had been given the rule

T → abcbbca

the third generation would have 42 more characters.

Undefined variables used as placeholders need not be added to L-Systems; values for them can be provided during interpretation.

Base L-Systems

Another use for undefined variables is in designing *base* L-Systems that can be supplemented by definitions for undefined variables.

Consider the previous L-System supplemented by a definition for T:

seed: S
rules: S → STS
T → aSb

Successive generations are:

S
STS
STSaSbSTS
STSaSbSTSaSTSbSTSaSbSTS
…

On the other hand, with a different definition for T, as in

seed: S
rules: S → STS

April 3, 2004; revised August 1, 2004

T → SabS

successive generations are

```
S
STS
STSSabSSTS
STSSabSSTSSTSabSTSSTSSabSSTS
        …
```

Although the generations of these two L-Systems are different, they both reflect the common part of their base L-System.

Base L-Systems can be used as a tool for designing L-Systems incrementally by giving them variable definitions..

References

1. *Designing with L-Systems, Part 1: String Rewriting Systems*, 2004: http://cs.arizona.edu/patterns/weaving/webdocs/gre_ls01.pdf

2. *Designing with L-Systems, Part 2: A Side Trip to Graphics*, 2004: http://cs.arizona.edu/patterns/weaving/webdocs/gre_ls04.pdf

Ralph E. Griswold
Department of Computer Science
The University of Arizona
Tucson, Arizona