

# Ginger: Implementing a new Lisp family syntax

James Dean Palmer  
Northern Arizona University  
Flagstaff, Arizona  
James.Palmer@nau.edu

## ABSTRACT

In this paper we introduce G-expressions, a new syntax based on the S-expression syntax utilized by most Lisp family languages. We have implemented a new homoiconic language, Ginger, based on this syntax and a Smalltalk inspired object system. Like the Scheme language, Ginger employs only a few special forms and observes a minimalist discipline allowing users to define functions that act like the primitive forms in many Algol-like languages. But unlike Scheme, G-expressions allow Ginger to emulate the aesthetic feel of an Algol-like language syntax. While fundamentally a dialect of Lisp, Ginger implements an attractive modern syntax which can superficially resemble Python or Ruby. This syntactic flexibility exemplifies Ginger's true power as a tool for developing task or domain-specific micro-languages.

## Categories and Subject Descriptors

D.3.1 [PROGRAMMING LANGUAGES]: Formal Definitions and Theory

## 1. INTRODUCTION

S-expressions are a syntax for representing complex hierarchical data structures. This syntax is most well known for providing the syntactic structure for Lisp, Scheme and other functional languages. S-expressions are composed of symbols, strings, literals, numbers and other S-expressions. Parentheses are used to denote the beginning and end of lists made up of elements separated by white space. Consider this S-expression example:

```
1 (("Lyra" (("Will" ("Roger" (a b)))) (1.73))
```

The expression represents a list with two members. Each member represents another list or S-expression. While essentially hierarchical it can be difficult to determine hierarchical relationships without counting parenthesis or using an editor that supports parenthesis matching.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACMSE '09 March 19-21, 2009, Clemson, SC  
©2009 ACM 978-1-60558-421-8/09/03 ...\$10.00

**Figure 1: A Koch snowflake is easily constructed from three Koch curves. This figure was generated using Ginger's turtle graphics library which closely mimics Logo's famous turtle.**

In this paper we describe G-expressions, an S-expressions variant with special rules that incorporate indentation and line continuations to simplify hierarchical notation and improve consistent readable formatting.

An alternate G-expression based rendering of the last example might read:

```
1 "Lyra"  
2 "Will"  
3 "Roger"  
4 a b  
5 1.73
```

The horizontal spacing replaces and augments the hierarchical structure provided by S-expression parenthesis. In this example it is much more apparent that "Lyra" and 1.73 are at the same hierarchical depth.

G-expressions are not a true superset of S-expressions but the first example happens to also be a valid G-expression. Experience has shown that S-expressions can be difficult for humans to decipher without adding extra formatting and white space to emphasize hierarchical relationships. G-expressions seek to take advantage of the natural organizational spacing most people add to S-expressions. The intent is to reduce parenthesis which are made redundant by explicit formatting choices. While many S-expression and G-expression analogs exist, G-expressions are not simply syntactic sugar for S-expressions. The semantics of structures formed from parenthesis and indentation is different and corresponds to different, but related, internal representations.









