A Grammar Notation for Tree Languages

Breck Yunits

***Abstract*—I introduce the core idea of a new grammar notation for formally describing Tree Languages.**

I. Introduction

Creating a great programming language is a multi-step process. One step in that process is to decide on syntax and formally define a language in a grammar notation such as BNF. Unfortunately, like the programming languages they describe, these grammar notations are complex and error-prone.

Below I introduce the core idea of a much simpler grammar notation for defining Tree Languages.

II. A Grammar Notation for Tree Languages

A Tree Language Grammar is a *double* consisting of a set of Keyword Definitions and a catchall Keyword.

A Keyword Definition is a *double* consisting of a unique keyword identifier and a Grammar.

Everything is encoded in Tree Notation, hence the grammar notation itself is a Tree Language.

III. Example

A Tree Language Grammar file for an imagined Tree Language called Tally, with 2 possible recursive keywords {+, -} might look like this:

```
Tally
 catchAll error
 keywords
  error
  expression Tally
   words int+
  + expression
  - expression
```

A valid program in the Tally language defined by the file above:

```
+ 4 5
 - 1 1
```

IV. Conclusion and Future Work

The introduction above is minimal but shows the core idea: Tree Languages can be formally defined in a simple grammar notation that itself is a Tree Language.

Ohayo Computer has developed a compiler-compiler for these grammar files. Future publications and/or open source releases will delve into the additional features found in the compiler-compiler and its associated grammar.

Breck Yunits is a researcher at Ohayo Computer (breck@ohayo.computer)