A Grammar Notation for ETNs

Breck Yunits

***Abstract*—I introduce the core idea of a new grammar notation for formally describing programming languages that extend Tree Notation.**

I. Introduction

Creating a great programming language is a multi-step process. One step in that process is to 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 programming languages that Extend Tree Notation (ETNs).

II. A Notation for ETN Grammars

An ETN Grammar is a *single* consisting of a set of Node Type Definitions.

A Node Type Definition is a *double* consisting of a unique node type identifier and an ETN Grammar.

Everything is encoded in Tree Notation, hence the grammar notation itself is an ETN.

III. Example

An ETN Grammar file for an imagined ETN called Tally, with 2 possible recursive node types {+, -} might look like this:

```
Tally
 + Tally
 - Tally
```

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: ETNs can be formally defined in a simple grammar notation that itself is an ETN.

Ohayo Computer has developed a feature-rich 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 ETN.

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