Zebu Specification

April 12, 2015

1 Overview

Zebu is meant to be a parser generator for the Go language, with it's primary focus being the ability to extend other grammars written in Zebu.

Yacc/Bison, ATNLR, and PPG (Polyglot Parser Generator) serve as inspirations for how Zebu should function, with features pulled from each.

1.1 Why Zebu?

Yacc/Bison laid the groundwork and serve as the model for compiler compilers in the field. Although there are implementations in Go, the syntax is a little outdated and does not feature the main motivation behind Zebu, the ability to extend grammars.

ANTLR serves as a great model for the syntactic design of a modern compiler compiler, but again is missing (and was not designed) the ability to extend grammars. This is not a flaw in ANTLR, but a unique need that requires a specific implementation.

PPG has most of the features that are required to extend a grammar (as that is what it is designed for) but lacks some of the power that is needed to fully extend a grammar.

Zebu is needed to resolve the three above compiler compilers into one that serves a specific purpose, the ability to fully extend grammars.

1.2 Goals

The main goal of Zebu is to generate extensible grammars, as such there will be design tradeoffs versus other compiler compilers.

Zebu highlights...

- Combine lexicon and grammar into one definition similar to ANTLR. The lexicon will be defined with regular definitions. The parser should be defined with a context free grammar.
- Unlike ANTLR, only regular definitions can serve as terminal symbols in the grammar. This is to enforce a consistent style (no mixing regular expressions, regular efinitions, and grammar rules).
- Regular definitions as well as grammar rules will be subject to the same extensible syntax and semantics. This allows extending grammars to override not only the semantics of a grammar, but the syntax as well.
- Generate LL(1) recursive decent parsers that can handle direct and indirect left recursion. The generated parser should be readable by humans.
- No global variables, Zebu should expose a parser object and mechanisms to modify this object. The API Zebu exposes should be operations on this object.
- No parse tree generation. ANTLR parse tree is very useful, but eventually an AST needs to be generated, and thus we have to revert to semantic actions anyways.

2 Lexical Analysis

3 Parser Generator