Contents

Chapter 1

Introduction

1.1 Background

XXX

1.2 Problem Description

XXX

1.3 Motivation

XXX

Chapter 2

Methodology

2.1 Understanding the Current Input Instructions

XXX Current input instructions for NJOY.

2.2 Designing the New Input Format

XXX define a grammar for the new input format?

2.2.1 Writing the Grammar

XXX

2.2.1.1 Scope and Structure

- static scope? only top-level declarations?
- block structure? e.g. `module { variable-declarations }`

2.3 Building the Compiler

2.3.1 Lexical Analysis

XXX

- Use a lexical analyzer to generate tokens and detect errors?

2.3.1.1 Tokens

XXX *<token-name, attribute-value>*

TOKEN	DESCRIPTION	EXAMPLE
module		`reconr`
number		`3.14`

2.3.1.2 Patterns

XXX should patterns be described?

2.3.1.3 Lexemes

XXX should lexemes be described?

2.3.1.4 Comments

XXX investigate the possibility to use comments in the input?

2.3.2 Syntactic Analysis

XXX

- Use a parser generator such as YACC which can detect and report syntax errors?

2.3.3 Semantic Analysis

XXX

- Enforce a type system? Type and scope rules defined by grammar.
- Detect and report errors?

2.3.4 Translation Process

XXX Code generation. No intermediate code necessary.

- process of translating the *source* language (new input format) into the *target* language (old input format)?
- detect and report errors?

Chapter 3

Implementation

XXX Python, {functional, object oriented} approach?

3.1 NJOY Input Format (`NIF`)

- Describe design?
- Grammar Definition File?

3.2 NJOY Input Format Translator (`nifty`)

- Python Lex?
- Python YACC?

3.2.1 Code Generation

XXX generating the *target* language instructions?

Chapter 4

Result

4.1 XXX

XXX

Chapter 5

Discussion

5.1 XXX

XXX

Chapter 6

Conclusion

- Improvement on the existing situation?

6.1 Future Work

- Possible improvements?
- Complete context-free grammar?
- GUI?
- Efficient implementation, e.g. C with pthreads?

Bibliography

[1] XXX