Contents

Chapter 1

Introduction

1.1 Background

The NJOY Nuclear Data Processing System [1] is a software package used for nuclear data management. In particular, it is used to convert Evaluated Nuclear Data Files (ENDF) [2] into different formats, as well as performing operations on the nuclear data.

NJOY is currently being used within the MACRO project [3] at the Division of Applied Nuclear Physics, at the Department of Physics and Astronomy at Uppsala University.

1.2 Problem Description

The NJOY input instructions are complex and hard to read compared to e.g. a high-level programming language. The listing below is a *short* and *simple* NJOY job which illustrates what the input instructions look like.

Algorithm 1.1 NJOY Test Problem 14

```
1  acer
2  20 21 0 31 32
3  1 0 1/
4  'proton + 7-n-14 apt la150 njoy99 mcnpx'/
5  725 0./
6  /
7  /
8  acer
9  0 31 33 34 35
10 7 1 2/
11 'proton + 7-n-14 apt la150 njoy99 mcnpx'/
12 viewr
13 33 36/
14 stop
```

Without consulting the documentation, one might guess that line 4 and 11 are some kind of descriptive titles, which is correct. But it is not obvious that line 2 denotes input and output files (each number indicates a specific file) which the `acer` module will operate on. It is also hard to deduce that the first number on line 5 denotes the material to be processed, and that that the second number denotes the desired temperature in kelvin.

This is not an optimal input format. The input instructions can be annotated with descriptive comments, but even then, working with a large and complex job easily becomes a daunting task.

1.3 Objective

The scope of this work has been to design and implement a more user friendly, and readable input format.

- E.g. denote scope, declaration of variables, et cetera.
- Resemble a high-level programming language?

Chapter 2

Methodology

2.1 Understanding the Current Input Instructions

XXX Current input instructions for NJOY.

- Complex
- Strict sequential order
- Terminology: input deck, decks of cards

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 Translator

XXX Python, UN*X.

2.3.1 Lexical Analysis

XXX

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

2.3.1.1 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?
- Building an AST?

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?
- Traverse the AST and output NJOY input instructions.

2.4 Testing

XXX Continuously testing during all phases.

- Compare output from `nifty` with expected output.

Chapter 3

Results

3.1 NJOY Input Format (`NIF`)

- Describe design?
- The Grammar?

3.2 NJOY Input Format Translator (`nifty`)

- PLY
  - Lex
  - YACC
- Translation of AST?
  - Code generation: generating the target language instructions?

Chapter 4

Discussion

- Testing not that rigorous. NJOY is a large and complex program, there's a lot of scenarios (e.g. input instructions) that hasn't been tested.
- Efficiency?
- Answer the "Why?" questions.
- Significant findings?

Chapter 5

Conclusion

- Improvement on the existing situation?
- Challenges?
  - Constructing a decent input format.
  - The physics (even if it's not within the scope). Documentation is full of it, kind of.
- Usable?
  - Production: not adviceable. Grammar not verified. "Toy" translator.

5.1 Future Work

- Possible improvements?
- Context-free grammar?
- GUI for `viewr` and `plotr`?
- Efficient implementation, e.g. C?

Bibliography

[1] http://t2.lanl.gov/codes/njoy99/

[2] http://www.nndc.bnl.gov/endf/

[3] MAssive Computation methodology for Reactor Operations (http://www.fysast.uu.se/tk/en/content/macro-project)