Chapter 1

Dirac Programming Language

Luis Felipe Gomes da Silva

Abstract

This document is about 'Dirac' programming language.

Resumo

Este documento é sobre a linguagem de programação Dirac.

1.1. General Information

1.1.1. Motivation

Dirac is a programming language that contains all the math required in Quantum Physics. Using Paul Dirac's *bra-ket* notation, it is possible to simulate some Quantum Calculations and by the use of random variables and vectors through pseudo random processes, to simulate very roughly, the uncertainty required in a physical system.

Furthermore, I'm working on a kind theory of a computational space, that is nothing more than a two dimension vector field that'll be able, as I hope, to eliminate time from computing, getting rid of those complex classes hierarchies.

1. Dirac is a stack based compiled language.
2. It is very rigorous and does not allow recursion.
3. It is an open source language released under GPL license.
4. It is fitted to science purposes only.
5. Any military use is condemned by the author.

Table 1.1. Table of Types

Symbols	Indexes	Numbers	Objects
program, end, ket, bra, is variable, constant, interval, integer, natural, rational, irrational,real, complex, ’!’, ’.’ , ’,’	variable, constant	interval, integer, natural,rational,irrational, complex, real	string, vector, matrix

[a] Comments are given by the token ’!’

1.1.2. Objectives

1. To represent the mathematics of quantum physics using Dirac’s bra-ket notation as atoms.
2. To compile code to MIPS architecture.

Due to the crescent monopoly on compilers made by the industry, I decided to create a language and a compiler to fit my needs.

1.2. First Stage

So far I have established the tokens I judge to be necessary for the first part. The preprocessor remains halted, because it is very difficult to know beforehand what we need as macros to prepare the code. Instead I prepared a simple scanner. The first part is concerned with these:

1. scan the stream for tokens.
2. attach the lexemas (types) to the tokens and save them in a table.

I turned the language more rigorous and more close to human understanding. Although it has the cost of productivity (which is not my aim), the language seems to become more readable and intuitive for mathematicians and physicists. Besides, languages which are interpreted and productive are messy and poorly documented, which annoys anyone who wants to grasp the mere elements of those languages. Here is a sample code for the first stage of development.

```
program, teste.
```

```
variable, a is bra.
variable, x is matrix.
transpose(a).
c := mult(a,x).
print(c).
end, program.
```

1.3. Second Stage

1.3.1. The Dirac's Grammar

Dirac Grammar's Structure