TUG 2022-online
July 22-24, 2022
luatruthtable package in LaTeX

Chetan Shirore[1] Dr. Ajit Kumar[2]

[1]KTHM College, Nashik, Maharashtra, India

[2]Institute of Chemical Technology, Mumbai, Maharashtra, India

Outline of Presentation

Introduction

The Lua programming language is a scripting language which can be embedded across platforms. With —*LuaTeX* and *luacode* packages, it is possible to use Lua in LaTeX. The TeX or LaTeX has scope for programming. However, with weird internals of TeX there are several limitations especially for performing calculations on numbers in LaTeX document. There are packages like *pgf* and *xparse* in LaTeX which provides some programming capabilities inside LaTeX documents. However such packages are not meant to provide the complete programming structure that in general other programming languages (like Lua) provide.The generation of Truth Tables with these packages in LaTeX get weird* and probably without using Lua it can't be done in an easier way in LaTeX. The programming capabilities of Lua are effectively used in the development of *luatruthtable*package. The *xkeyval* package is used in its development apart from the —*luacode* package. The time for generation of truth tables using the package and compilation of TeX document with LuaTeX is not an issue as Lua has dynamic memory management.

*Macro for automating truth tables in LaTeX
https://tex.stackexchange.com/questions/505994
visited on 2022-02-22

Installation, Inclusion and License of the Package

The installation of *luatruthtable* package is similar to plain latex package where `.sty` file is in LaTeX directory of texmf tree. The package can be used by including the command `\usepackage{luatruthtable}` in the preamble of latex document. The TeX file is to be compiled using the LuaLaTeX.

The *luatruthtable* package is released under the LaTeX Project Public License v1.3c or later. The complete license text is available at http://www.latex-project.org/lppl.txt. It is developed in lua. Lua is certified open source software. Its license is simple and liberal which is compatible with GPL.

Core Ideas used in the development of the package

The Lua is extensible language that can be embedded in LaTeX. The TeXlanguage has indirect support for scripting languages*. The function *toBinary(x,y)* is used to produce sequence of *True* and *False* values of boolean variables. This function converts the decimal number x to a binary number by adding y number of leading zeroes. The result of this is stored in a table in Lua. Here y corresponds to number of boolean variables. As 2^y permutations of boolean variables are to be produced, the function *toBinary(x,y)* runs inside a loop where x takes values from 1 to y. The splitting of variables and expressions is done using string methods available in Lua. The nested *metamethods* in Lua are used to define several logical operators. The *load* function in Lua is used to evaluate logical expressions.

*TEX and Scripting Languages
https://tug.org/TUGboat/tb25-1/richter.pdf
visited on 2022-03-01

luaTruthTable command in luatruthtable package

The `\luaTruthTable` command is the main command in the *luatruthtable* package which generates truth tables. It has the following syntax.

```
\luaTruthTable[trtext,fltext]
{list of boolean / logical variables}{list of  expressions}
```

The command has two mandatory arguments.
i) `list of boolean / logical variables` : The list of boolean or logical variables should be separated by comma.
ii) `list of expressions` : The list of logical expressions that are to be evaluated should be separated by comma.
The command has two optional arguments.
i) `trtext` : The `trtext` is the display value for the boolean value *True*. It has the default value `$T$` in the package. It can be any string or number. Assigning value 0 should be avoided to `trtext` variable.
ii) `fltext` : The `fltext` is the display value for the boolean value *False*. It has the default value `$F$` in the package. It can be any string or number. Assigning value 1 should be avoided to `fltext` variable.

Demonstrations

Let's see some demonstrations of commands in the package.

Known Issues,Limitations and Scope of the package

The associativity and precedence of operators is not yet supported. The package can give appropriate results only when parentheses are used for each of the operation. It gives erroneous result when parentheses are not used. This point is of utmost importance in using the package. There is no native way of defining a custom operator* in Lua. However some metamethods can be nested in a way to replicate an operator. All operators defined in this package are instances of such nesting. The question may be raised that are there better ways of accomplishing these in Lua. The answer is yes. The alternative ways may be better in some or other sense.

*Custom Operators in Lua

http://lua-users.org/wiki/CustomOperators

visited on 2022-03-02

Known Issues,Limitations and Scope of the package

For example, instead of defining **logand** operator and using it in the fashion *p*logand*q*, one could define function *logand* that takes two arguments and use it in a way *logand(p,q)*. But when it comes to embedding it in LaTeX, one has to use more and more nested parentheses as number of statements and operations increase. This is the exact reason why such approach is not used in the development of package. Instead of using *implies(logand(p,logor(q,r)),s)* it sounds more natural to use *(p *logand* (q*logor*r))*implies* s*.

Apart from these issues, there is no error handling mechanism used in the package. It relies on the error handling of Lua and TeX itself. The package currently supports 9 operations viz. *not*,*and*,*nand*,*or*, *xor*, *implies*, *iff*, *nor*, *xnor*. The error handling and extending number of operations may be considered in future versions of the package.

Thank You...!!!