The luatruthtable Package

Chetan Shirore* and Ajit Kumar

November 17, 2023

1 The luatruthtable Package

The `luatruthtable` package is developed to generate Truth Tables of boolean values in LaTeX. It provides an easy way for generating truth tables in LaTeX . There is no need of special environment in the package for generation of Truth Tables. It is written in lua and tex file is to be compiled with LuaLatex engine. The time required for operations is no issue while compiling with LuaLaTeX. There is no need to install lua on users system as tex distributions (TeXLive or MikTeX) come bundled with LuaLaTeX. It is useful for generation of Truth Tables in tex file itself. The package supports nesting of commands for multiple operations. It can be modified or extended by writing custom lua programs.

The programming capabilities of Lua are effectively used in the development of pacakage. The other approaches of generating Truth Table in LaTeX get weired [2] and probably without using Lua it can't be done in an easier way in LaTeX. The *xkeyval* package is used in its development apart from the *luacode** package.

2 License

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 available. Its license is simple and liberal which is compatible with GPL.

3 Installation and Inclusion

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 `\luatruthtable` in the preamble of latex document. The tex file is to be compiled using the LuaLaTeX engine.

4 Core Ideas used in the development of the package

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.

*Email id: mathsbeauty@gmail.com

5 The luaTruthTable command

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 a comma.
ii) `list of expressions` : The list of logical expressions that are to be evaluated should be separated by a 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.

The `\luaTruthTable` command should be used within in the environment `\begin{tabular} ... \end{tabular}` or any other environment in LaTeX for tables. The sequence of column heads should be same as sequence of `list of boolean/logical variables` and `list of expressions`. Without these correct inputs, the command `\luaTruthTable` can not produce the desired output.

6 Operations in the luatruthtable package

a) *not*: The value of *not p* is False when p is True and it is True when p is False.

p	not p
T	F
F	T

The command `lognot*` is used in the package to generate truth table for *not* operation.

b) *and*: The value of p *AND* q is True if and only if both p and q are True.

p	q	p and q
T	T	T
T	F	F
F	T	F
F	F	F

The command `*logand*` is used in the package to generate truth table for *and* operation .

c) *or*: The value of p *or* q is False if and only if both p and q are False.

p	q	p or q
T	T	T
T	F	T
F	T	T
F	F	F

The command `*logor*` is used in the package to generate truth table for *or* operation.

d) *implies*: The value of p *implies* q is False if and only if p is True and q is False.

p	q	p implies q
T	T	T
T	F	F
F	T	T
F	F	T

The command `*imp*` is used in the package to generate truth table for *implies* operation.

e) *if and only if* : The value of p *if and only if* q is True if and only if both p and q have same truth values.

p	q	p iff q
T	T	T
T	F	F
F	T	F
F	F	T

The command `*iff*` is used in the package to generate truth table for *if and only if* operation.

f) *NAND* : The value of p *NAND* q is 0 if and only if both p and q have value 1.

p	q	p NAND q
1	1	0
1	0	1
0	1	1
0	0	1

The command `*lognand*` is used in the package to generate truth table for *NAND* operation.

g) *XOR* : The value of p *XOR* q is 0 if and only if p and q have same values.

p	q	p XOR q
1	1	0
1	0	1
0	1	1
0	0	0

The command `*logxor*` is used in the package to generate truth table for *XOR* operation.

h) *NOR*: The value of p *NOR* q is 1 if and only if both p and q have value 0.

p	q	p NOR q
1	1	0
1	0	0
0	1	0
0	0	1

The command `*lognor*` is used in the package to generate truth table for *NOR* operation.

i) *XNOR*: The value of p *XNOR* q is 1 if and only if both p and q have same values.

p	q	p XNOR q
1	1	1
1	0	0
0	1	0
0	0	1

The command `*logxnor*` is used in the package to generate truth table for *XNOR* operation.

Table 1 describes operators in the `luatruthtable` package.

Operator	Syntax	Description
`lognot*`	`lognot*p`	Negates the boolean variable p.
`*logand*`	`p*logand*q`	Gives the truth value of the expression p *and* q.
`*logor*`	`p*logor*q`	Gives the truth value of the expression p *or* q.
`*imp*`	`p*imp*q`	Gives the truth value of the expression *if* p *then* q.
`*iff*`	`p*iff*q`	Gives the truth value of the expression p *if and only if* q.
`*lognand*`	`p*lognand*q`	Gives the truth value of the expression p *NAND* q.
`*logxor*`	`p*logxor*q`	Gives the truth value of the expression p *XOR* q.
`*lognor*`	`p*lognor*q`	Gives the truth value of the expression p *NOR* q.
`*logxnor*`	`p*logxnor*q`	Gives the truth value of the expression p *XNOR* q.

Table 1: Operators in the luatruthtable package

7 Illustrations of commands in the luatruthtable package

The *luatruthtable* package can accept a finite number of variables. It supports a finite number of variables that one would need.

Listing 1: LaTeX document with luatruthtable package

```
\begin{tabular}{|cc|c|c|c|c|c|c|c|c|c|}
\hline
\(p\) & \(q\) & \(\neg p\) & \(p \land q\) & \(p\lor q\) & \(p \rightarrow q\)
 & \(p \leftrightarrow q\) & \( p\) nand \(q\) & \(p\) xor \(q\) & \( p\) nor \(q\) &
    \(p\) xnor \(q\) \\
\hline
\luaTruthTable{p,q}{lognot*p, p*logand*q, p*logor*q, p*imp*q, p*iff*q,
p*lognand*q, p*logxor*q, p*lognor*q, p*logxnor*q } \\
\hline
\end{tabular}
```

The LaTeX code (Listing 1) generates the output shown in Table 2.

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	p nand q	p xor q	p nor q	p xnor q
T	T	F	T	T	T	T	F	F	F	T
T	F	F	F	T	F	F	T	T	F	F
F	T	T	F	T	T	F	T	T	F	F
F	F	T	F	F	T	T	T	F	T	T

Table 2: Example 1

Listing 2: LaTeX document with luatruthtable package

```
\begin{tabular}{|cccc|c|}
\hline
\(p\) & \(q\) & \( r\) & \(s\) & \(((p\land q) \rightarrow \neg r) \land s \) \\
\hline
\luaTruthTable{p,q,r,s}{((p*logand*q)*imp*(lognot*r))*logand*s } \\
\hline
\end{tabular}
```

The LaTeX code (Listing 2) generates the output shown in Table 3.

p	q	r	s	$((p \wedge q) \rightarrow \neg r) \wedge s$
T	T	T	T	F
T	T	T	F	F
T	T	F	T	T
T	T	F	F	F
T	F	T	T	T
T	F	T	F	F
T	F	F	T	T
T	F	F	F	F
F	T	T	T	T
F	T	T	F	F
F	T	F	T	T
F	T	F	F	F
F	F	T	T	T
F	F	T	F	F
F	F	F	T	T
F	F	F	F	F

Table 3: Example 2

8 Known Issues, Limitations and Scope of the package

The associativity and precedence of operator is not yet supported. The package can give appropriate results only when parentheses are used for each of the operation. It gives eraneous result when parentheses are not used. This point is of utmost importance in using the package. These issues are there is no native way of defining custom operations in Lua [1]. However some metamethods can be nested in a way to glimpse the operators. All operators defined in this package are instances of such nestings. The question may be raised that is there better way of accomplishing this in Lua. The answer to the point is that there are alternative

ways of doing this. They may be better in some or other sense. 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, again 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.

References

[1] *Lua Custom Operator*. visited on 2022-02-22. URL: http://lua-users.org/wiki/CustomOperators.

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