The `luatruthtable` LaTeX package

Chetan Shirore, Ajit Kumar

Abstract

This paper describes the development and usage of the *luatruthtable* package in LaTeX. It is developed to generate truth tables of boolean values in LaTeX documents. The package provides an easy way of generating truth tables in LaTeX. There is no need of a special LaTeX environment for generation of truth tables with the package. It is written in Lua and the TeX document is to be compiled with the LuaLaTeX engine.

1 Introduction

The Lua [1] programming language is a scripting language which can be embedded across platforms. With LuaTeX [4], it is possible to use Lua in LaTeX. TeX or LaTeX has scope for programming in themselves. However, with the internals of TeX there are several limitations especially for performing calculations on numbers in LaTeX documents. There are packages like *pgf* [5] and *xparse* [9] in LaTeX which provide 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 gets complicated [7] and probably without using Lua it can't be done in an easier way in LaTeX. The programming facilities of Lua are effectively used in the *luatruthtable* package. The *xkeyval* package is used in its development, in addition to the *luacode* package [2]. The time for generation of truth tables using the package and compilation of TeX document with LuaTeX is not an issue.

2 Installation and inclusion

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

3 Core ideas used in the development of the package

Lua [1] is an extensible language that can be embedded in LaTeX. The TeX [8] language has indirect support for scripting languages [6].

The *luatruthtable* function `toBinary(x,y)` is used to produce a sequence of *True* and *False* values

doi.org/10.47397/tb/43-2/tb134shirore-luatruthtable

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 the 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.

4 The `\luaTruthTable` command in the `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 commas.

ii) ⟨*list of expressions*⟩: The list of logical expressions that are to be evaluated should also be separated by commas.

And the command has two optional arguments:

i) ⟨*trtext*⟩: the display value for the boolean value *True*. It has the default value `$T$` in the package. It can be any string or number, although assigning the value 0 would not make sense.

ii) ⟨*fltext*⟩: the display value for the boolean value *False*. It has the default value `$F$` in the package. It can be any string or number, although assigning the value 1 would similarly not make sense.

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

5 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 a truth table for the *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
F	T	F
T	F	F
T	T	T
F	F	F

The command `*logand*` is used in the package to generate a truth table for the *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
F	T	T
T	F	T
T	T	T
F	F	F

The command `*logor*` is used in the package to generate a truth table for the *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
F	T	T
T	F	F
T	T	T
F	F	T

The command `*imp*` is used in the package to generate a truth table for the *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
F	T	F
T	F	F
T	T	T
F	F	T

The command `*iff*` is used in the package to generate a truth table for the *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
0	1	1
1	0	1
1	1	0
0	0	1

The command `*lognand*` is used in the package to generate a truth table for the *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*
0	1	1
1	0	1
1	1	0
0	0	0

The command `*logxor*` is used in the package to generate a truth table for the *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*
0	1	0
1	0	0
1	1	0
0	0	1

The command `*lognor*` is used in the package to generate a truth table for the *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*
0	1	0
1	0	0
1	1	1
0	0	1

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

Table 1 summarises logical operators in the package.

Table 1: Operations in the `luatruthtable` package, given boolean variables *p* and *q*.

Command	Description
`lognot*p`	Negates the boolean variable *p*.
`p*logand*q`	Truth table for the expression *p and q*.
`p*logor*q`	Truth table for the expression *p or q*.
`p*imp*q`	Truth table for the expression *if p then q*.
`p*iff*q`	Truth table for the expression *p if and only if q*.
`p*lognand*q`	Truth table for the expression *p NAND q*.
`p*logxor*q`	Truth table for the expression *p XOR q*.
`p*lognor*q`	Truth table for the expression *p NOR q*.
`p*logxnor*q`	Truth table for the expression *p XNOR q*.

6 Examples and usage

The *luatruthtable* package accepts a finite number of variables. It supports any finite number of variables that one would need in practice. A few examples of usage are given here.

The following example involves three variables, `p`, `q`, and `r`.

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

The output from the above is shown in Table 2.

Here *lognot*r* is enclosed in parentheses to produce correct results in the generated truth table.

The following is the code generated by the command `\luaTruthTable` in the above code.

```
$F$ & $F$ & $T$ & $T$\\
$F$ & $T$ & $F$ & $T$\\
$F$ & $T$ & $T$ & $T$\\
$T$ & $F$ & $F$ & $T$\\
$T$ & $F$ & $T$ & $T$\\
$T$ & $T$ & $F$ & $T$\\
$T$ & $T$ & $T$ & $F$\\
$F$ & $F$ & $F$ & $T$
```

With the use of optional arguments *[trtext=True, fltext=False]* in the previous example, one gets the following output:

p	*q*	*r*	$(p \land q) \rightarrow \neg r$
False	False	True	True
False	True	False	True
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	False
False	False	False	True

Table 2: Example output from the `\luaTruthTable` command.

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

It is possible to give *trtext* and *trfalse* values that are TEX math text. So with the use of optional arguments `[trtext=$T$, fltext=$F$]` in the previous example, one gets the following output:

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

Since the luacode* environment is used, the backslash is to be escaped in setting *trtext* and *trfalse*. For example: `[trtext=\\(True\\), fltext=\\(False\\)]`.

7 Known issues, limitations and scope of the package

The associativity and precedence of operators is not yet supported. Thus the package can give appropriate results only when parentheses are used for each of the operations, and gives erroneous results 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 [3]. 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 one way or another. 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 in LATEX, one has to use more and more nested parentheses as the number of statements and operations increases. This is the exact reason why this approach was not used in the development of the package. Instead of using *implies*(*logand*(*p*,*logor*(*q*,*r*)),*s*) it seems more natural to use (*p *logand* (q*logor*r)*)**implies* s*.

Also, 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 the nine listed operations, viz. *not, and, nand, or, xor, implies, iff, nor, xnor*. Error handling and extending the number of operations may be considered in future versions of the package.

The package, including source code, is released through CTAN (`ctan.org/pkg/luatruthtable`), and in the usual TEX distributions.

References

[1] Lua 5.4 reference manual. `https://www.lua.org/manual/5.4`

[2] Luacode package page, 2012. `https://ctan.org/pkg/luacode`

[3] Lua custom operators. `http://lua-users.org/wiki/CustomOperators`

[4] Luatex package page. `https://ctan.org/pkg/luatex`

[5] PGF package page. `https://ctan.org/pkg/pgf`

[6] W.M. Richter. TEX and scripting languages. *TUGboat* 25(1):71–88, 2004. `https://tug.org/TUGboat/tb25-1/richter.pdf`

[7] StackExchange. Macro for automating truth tables. `https://tex.stackexchange.com/questions/505994`

[8] Wikipedia. TeX. `https://en.wikipedia.org/wiki/TeX`

[9] Xparse package page. `https://ctan.org/pkg/xparse`

◇ Chetan Shirore
Department of Mathematics,
K.T.H.M. College, Nashik
422002, Maharashtra, India
`chetanshirore (at) kthmcollege dot ac dot in`

◇ Ajit Kumar
Department of Mathematics,
Institute of Chemical Technology, Mumbai
400019, Maharashtra, India