

truthtable: L^AT_EX Package for automatically generated Truth Tables

Dominic Flück <K-Trout>

0.0.2 2021/10/01

Abstract

`truthtable` is a L^AT_EX package for creating automatically generating truth tables given a table header. It supports a number of logical operations which can be combined as needed. It's built upon the package `luacode` and therefore has to be used with the LuaL^AT_EX compiler.

Contents

1	Introduction	1
2	Dependencies	2
3	Usage	2
3.1	Comma separated variables	2
3.2	Comma separated display variables	2
3.3	Comma separated statements	2
3.3.1	NOT / Negation	2
3.3.2	AND / Conjunction	2
3.3.3	OR / Disjunction	3
3.3.4	XOR / Exclusive disjunction	3
3.3.5	NAND / Negated conjunction	3
3.3.6	→ / Implication	3
3.3.7	↔ / Equality	3
3.4	Comma separated display statements	3
3.5	Display true value	3
3.6	Display false value	4
4	Example of use	4
5	Development	5
5.1	Repository	5
5.2	Changes	5
5.3	Known issues and bugs	5

1 Introduction

Tables in L^AT_EX have the reputation of being a bit tedious. When creating a table with many cells, such as a truth table, they are not only tedious to build, but also not very readable.

To help this situation when creating a truth table for a document, this package provides a macro, which allows simply for the variables and the columns of a truth table to be defined. The package then takes care of the rest.

2 Dependencies

`truthtable` uses the package `luacode` to run, as the heavy lifting of the processing is done in *Lua*. The package checks if `luacode` is already loaded, and if not, does so. Lua_{TEX} is required to compile the resulting documents.

3 Usage

The `truthtable` package provides as of this version a single command:

```
\truthtable{comma separated variables}{comma separated display variables}  
{comma separated statements}{comma separated display statements}  
{display true value}{display false value}
```

The command positions in the normal table boilerplate. This leads to the redundant practice of defining the column count twice, once for the table environment as the column layout and once in the command by defining the variables and statements.¹

This is intentional to allow for more flexibility in customising the column layout as well as pre- and appending of further rows to the table.

3.1 Comma separated variables

The basic variables, for which every combination of *true* and *false* a row of table will be generated. The variables should be relatively simple, as they are not used for the formatting the table but simply to calculate the answers. The variables should be separated using commas. Don't use variables, which contain another variable, i.e., don't do this: `{n,An}`.

3.2 Comma separated display variables

These are the display values corresponding to the *Comma separated variables*. Fancy variable formatting can be applied. At least normal text and “math” mode seem to work.² The same number of display variables as variables is required. The comma cannot be used as a display character, as it is used as delimiter.

3.3 Comma separated statements

The statements using the *Comma separated variables* which are used to evaluate the statements for any given combination of variables. Parentheses can be used in the normal fashion to indicate the order of combined statements. The notation for the different operations is as follows:

3.3.1 NOT / Negation

To negate a variable or statement, the exclamation point `!` is used.

- $\neg A$: `!A`
- $\neg(\neg A)$: `!(!A)`

3.3.2 AND / Conjunction

For the conjunction of two variables or statements the and symbol `&` is used. **The `&` must not be escaped for the comma separated statements!**

- $A \wedge B$: `A & B`
- $A \wedge (A \wedge B)$: `A & (A & B)`

¹See [Listing 1](#) for example

²More testing needs to be done

3.3.3 OR / Disjunction

For the Disjunction of two variables or statements the vertical line character | is used.

- $A \vee B$: `A | B`
- $A \vee (A \vee B)$: `A | (A | B)`

3.3.4 XOR / Exclusive disjunction

The exclusive disjunction (XOR) is written in parentheses preceded by the hat operator. **Note that the delimiter used is the semicolon ; and not the comma , ! This is because the statements are separated using the comma.**

- $A \veebar B$: `^(A; B)`
- $A \veebar (A \vee B)$: `^(A; (A | B))`

3.3.5 NAND / Negated conjunction

The NAND operation is written in parentheses preceded by the the NOT and the AND operator (!&). **Note that the delimiter used is the semicolon ; and not the comma , ! This is because the statements are separated using the comma.**

- $A|B$: `!&(A; B)`
- $A|(A \vee B)$: `!&(A; (A | B))`

3.3.6 \rightarrow / Implication

The implication can also be expressed. **Note that the delimiter used is the semicolon ; and not the comma , ! This is because the statements are separated using the comma.**

- $A \rightarrow B$: `>>(A; B)`
- $A \rightarrow (A \vee B)$: `>>(A; (A | B))`
- $A \wedge (A \rightarrow B)$: `A & >>(A; B)`

3.3.7 \leftrightarrow / Equality

The equality can also be expressed. Since version 0.0.2 this command can also be expressed as `<>(A; B)`. The previous definition³ of `__(A; B)` also works. **Note that the delimiter used is the semicolon ; and not the comma , ! This is because the statements are separated using the comma. The __ must not be escaped for the comma separated statements!**

- $A \leftrightarrow B$: `__(A; B) = <>(A; B)`
- $A \leftrightarrow (A \vee B)$: `__(A; (A | B)) = <>(A; (A | B))`
- $A \wedge (A \leftrightarrow B)$: `A & __(A; B) = A & <>(A; B)`

3.4 Comma separated display statements

Display statements are defined the same way as the *comma separated display variables*. The comma cannot be used as a display character, as it is used as delimiter.

3.5 Display true value

The displaying string which will be used in the table body for *true*. Normal text and “math” mode can be used.

³The equality operation was defined this way in v0.0.1

3.6 Display false value

The displaying string which will be used in the table body for *false*. Normal text and “math” mode can be used.

4 Example of use

The code snippet seen in Listing 1 is the entirety of code required to produce the truth table seen in Table 1.⁴

The command generates the code seen in Listing 2.

Listing 1: Code for an sample truth table

```
\begin{table}[h]
\centering
\begin{tabular}{c|c||c|c|c|c|c|c|c}

% Content of table is generated using this single command.
\truthtable{A,B}{${A}$,$B$}
{!A, A & B, A | B, ~(A; B), !&(A; B), >>(A; B), <>(A; B)}{${\not A}$, $A \land B$, $A \lor B$
$, $A \veebar B$, $A | B$, $A \rightarrow B$, $A \leftrightarrow B$}
{${T}$}{${F}$}

\end{tabular}
\end{table}
```

Listing 2: Code generated by \truthtable

```
${A}$ & ${B}$ & ${\not A}$ & $A \land B$ & $A \lor B$ & $A \veebar B$ & $A | B$ & $A \rightarrow B$ & $A \leftrightarrow B$ \\
\hline
${T}$ & ${T}$ & ${F}$ & ${T}$ & ${T}$ & ${F}$ & ${T}$ & ${T}$ & \\
${T}$ & ${F}$ & ${T}$ & ${F}$ & ${T}$ & ${T}$ & ${T}$ & ${F}$ & \\
${F}$ & ${T}$ & ${T}$ & ${F}$ & ${T}$ & ${T}$ & ${T}$ & ${F}$ & \\
${F}$ & ${F}$ & ${T}$ & ${F}$ & ${F}$ & ${F}$ & ${T}$ & ${T}$ & \\\hline
```

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \veebar B$	$A B$	$A \rightarrow B$	$A \leftrightarrow B$
T	T	F	T	T	F	F	T	T
T	F	F	F	T	T	T	F	F
F	T	T	F	T	T	T	T	F
F	F	T	F	F	F	T	T	T

Table 1: Sample truth table

⁴The captioning setup was omitted in the listing.

5 Development

5.1 Repository

This package is on *CTAN* (ctan.org/pkg/truthtable). The repository of the package is github.com/K-Trout/truthtable. For bug reports and feature requests create an issue on github: github.com/K-Trout/truthtable/issues.

5.2 Changes

v0.0.2 (2021/10/TODO)

- Added support for *XOR* and *NAND*
- Added some error messages when the number of arguments and display arguments don't correspond.
- Added definition for equivalence operation to be written as $\langle \rangle (A; B)$. $_{_}(A; B)$ is still supported

v0.0.1 (2021/10/01)

- Initial release

5.3 Known issues and bugs

Stability The Lua code of the macro is not very error resistant. The package only checks if the same amount of working and display variables, as well as working and display statements. If a mismatch is detected, an error message is output and the package code halts. Further improvements may be undertaken in the future.

Display formatting Whilst normal text and “math” mode work for both headers and truth values, other text formatting such as `\textbf` does not. It is not yet clear if this will be addressed in future versions.

Operations For the moment seven operations are defined. Further operations may be added in future versions.