Boolean Syntax Extensions for the TI-Nspire CX CAS Handheld and Associated Emulators.

Language: English

Version: 1.0

Date: 6/6/2018

Author: James O. Thompson

Email: jodaddy101@hotmail.com

Table of Contents

1. Introduction

The functions presented here are not intended to teach Boolean algebra. Instead they should complement a good introductory text on the subject. Such a text can be found in “Finite Mathematics” by Lial, Greenwell and Ritchey. This book is frequently used in a college course for students majoring in business, management, economics, etc.

These functions are for the student equipped with a TI-Nspire CX CAS handheld or an emulator for one on a PC, iPad, etc. The “CAS” (Computer Algebra System) part is essential. The functions cannot be installed on the TI-Nspire without the CAS part.

The TI-Nspire has a rich set of Boolean operators. Unfortunately, these Boolean operators suffer in two aspects: they are far too verbose and they differ from that commonly used in textbooks for Boolean algebra.

For example, the textbook Boolean expression for the distributive law is concisely:

"q∧((q∧~p)∨p)⇔q∧((p∨q)∧(p∨~p))"

The equivalent TI-Nspire expression for this same law is:

"q and ((q and not p) or p)⇔q and ((p or q) and (p or not p))"

The TI-Nspire form, while fully as functional, is not as easily understood and is completely unusable for Boolean algebraic manipulations.

The functions packaged here augment (not replace) the existing operators and provide translations between the two syntaxes. Note that the augmentations are functional as well as cosmetic. The Boolean expressions may be executed with exactly the same results as the standard TI-Nspire operators.

Additionally, truth tables are a common need when studying Boolean algebra. There are functions in this package to easily generate truth tables using the Boolean augmentations. And, using truth table, functions are included to prove or disprove Boolean equations.

2. Platforms

The functions presented here will execute on the TI-Nspire CX CAS handheld, the TI-Nspire CX CAS Student Software and associated emulators executing on the iPad.

3. Installation

All of these functions are bundled into a single file, “boolean.tns”. This file is to be installed in the “mylib” folder where it will join the other files (numtheory.tns and linalgcas.tns) from the default installation. The file can be transferred to the handheld using the process described for file transfers in the handheld owner’s manual. On a PC the “mylib” subdirectory is in the User’s Documents subdirectory which, unfortunately, Microsoft likes to hide. See this web reference on how to locate the User’s Documents subdirectory.

After installing this file, you must “Refresh Libraries”. On the handheld, press Doc, then choose the “Refresh Libraries” option. On the Student Software, choose the Tools menu, then the “Refresh Libraries” option. Following this the functions may be accessed as the file name (boolean) followed by the backslash (\) followed by the function name. e.g., “boolean\truthtable()”.

Following installation, the function source code may be inspected and modified as desired. Furthermore, it may be distributed freely and for free.

4. Boolean Syntax Extensions

The syntax described here is a superset of standard TI-Nspire syntax, not a replacement. The Boolean expressions include all the standard stuff plus these syntax synonyms:

Standard	Augmentation	Description
“not”	“~”	Negation
“and”	“∧”	Conjunction
“or”	“∨”	Inclusive disjunction
“xor”	“⊙”	Exclusive disjunction
“nor”	“↓”	Joint denial
“nand”	“↑”	Alternative denial
“⇒”	“⇒”	Implication
“⇔”	“⇔”	Double implication

The augmentation symbols are self-delineating. That is, they do not require spaces before or after. It should be noted that braces {} and brackets [] are not the same as parenthesis (). They retain the standard meaning for lists and indices.

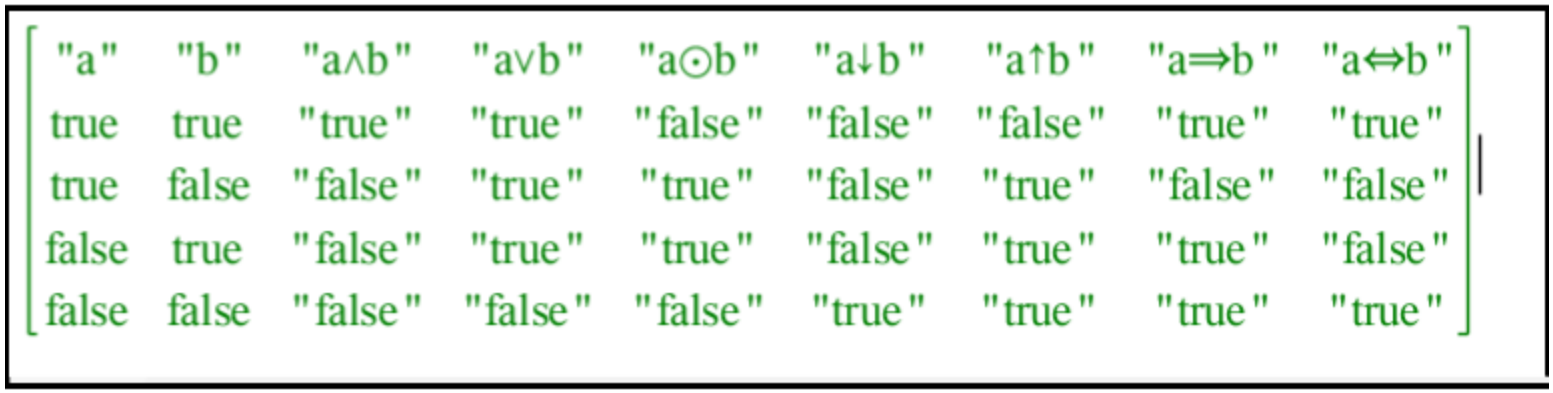
The augmented syntax will always be in a string to be interpretively executed by the functions in this package. The string may include standard expressions freely intermixed with the augmentation symbols. For example, the following is acceptable:

"q and ((q∧~p) or p)⇔q and ((p or q) and (p∨~p))"

Another example:

"√(b^2-4*a*c)>0∧bill∧home∨√(b^2-4*a*c)>0∧home∧joe"

Here is a truth table for the augmentation symbols:



"a"	"b"	"a∧b"	"a∨b"	"a⊙b"	"a↓b"	"a↑b"	"a⇒b"	"a⇔b"
true	true	"true"	"true"	"false"	"false"	"false"	"true"	"true"
true	false	"false"	"true"	"true"	"false"	"true"	"false"	"false"
false	true	"false"	"true"	"true"	"false"	"true"	"true"	"false"
false	false	"false"	"false"	"false"	"true"	"true"	"true"	"true"

5. Syntax Integration

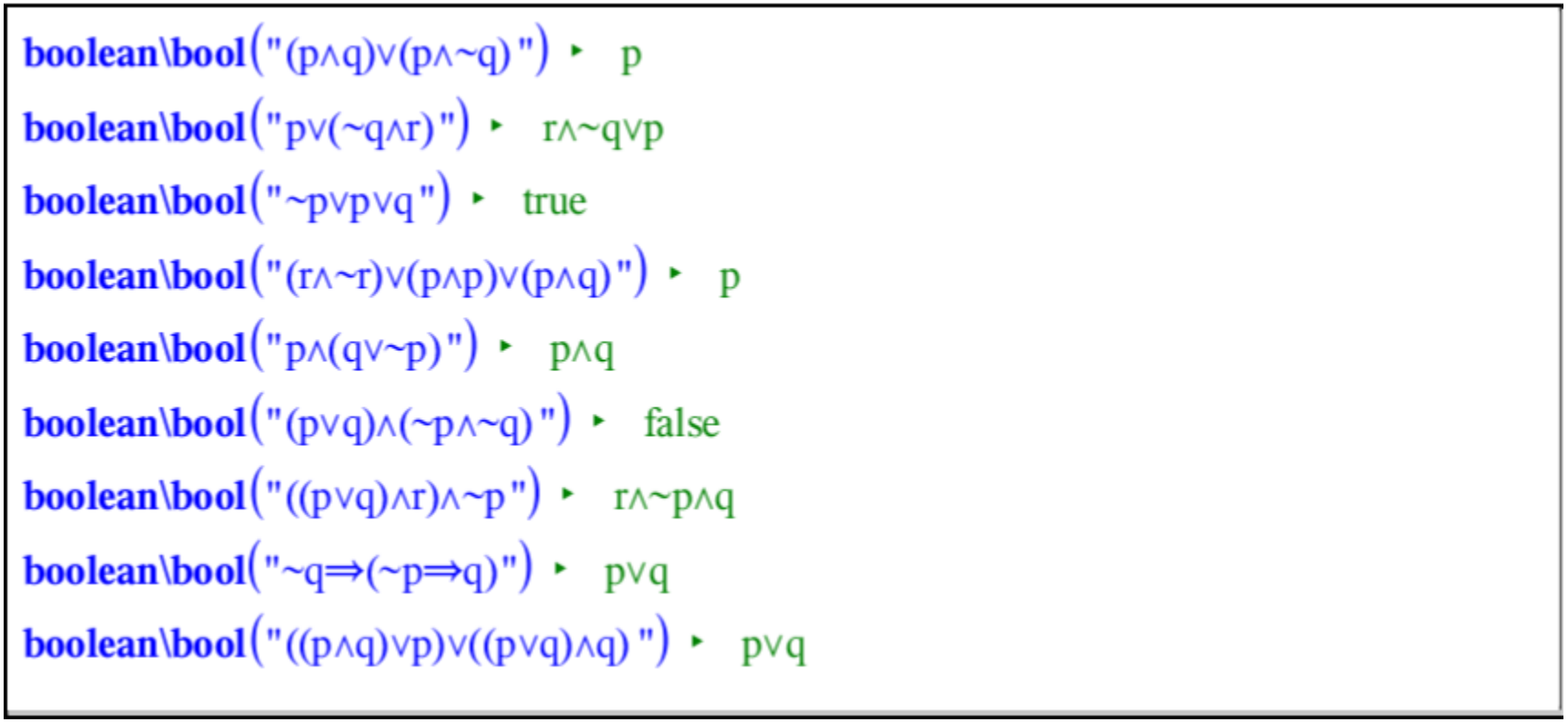
These functions implement the Boolean syntax described above. It should be noted that they provide no functionality beyond that already present in the TI-Nspire CX CAS handheld. They do provide a "wrapper" around that functionality.

a. boolean\bool()

This function accepts a string mathematical expression with the augmented Boolean syntax, translates it into a standard TI-Nspire expression, evaluates the expression using standard TI-Nspire facilities, captures the resultant expression and translates it back into augmented Boolean syntax.

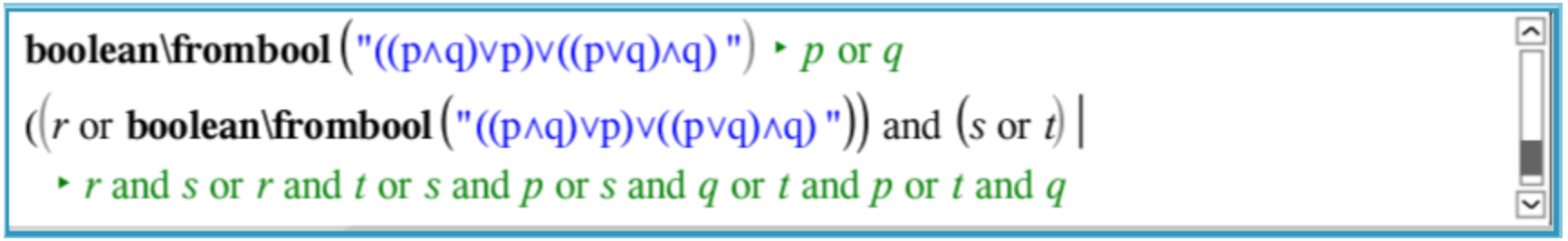
Note that, like standard TI-Nspire expressions, the variables may be symbolic, Boolean variables or constants, or even lists or matrices containing these elements.

The capabilities may best be seen by examples as the following:



b. boolean\frombool()

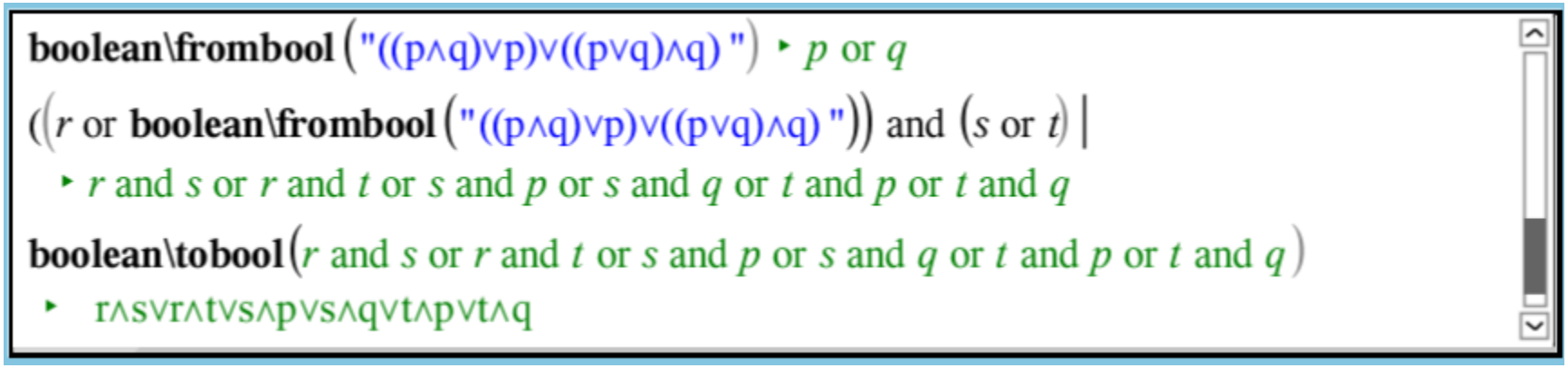
The bool() function above does not allow integration into standard TI-Nspire expressions. This function does provide that functionality. It accepts a string with augmented Boolean syntax and translates it into a standard TI-Nspire expression as the following example shows:



This function also illustrates just how "unreadable" Boolean functions are in standard TI-Nspire syntax.

c. boolen\tobool()

This function goes the other way, translating a standard TI-Nspire expression into a string with augmented Boolean syntax as the following example illustrates:



6. Truth Table Generation Functions

a. boolean\truthtable()

This function takes a string containing an augmented Boolean expression, extracts the variable names, constructs and populates

a truth table for the expression as shown by the following example:

```
boolean\truthtable("~(~p∧~q)∨(~r∨~s)")
```

"p"	"q"	"r"	"s"	"~(~p∧~q)∨(~r∨~s)"
true	true	true	true	"true"
true	true	true	false	"true"
true	true	false	true	"true"
true	true	false	false	"true"
true	false	true	true	"true"
true	false	true	false	"true"
true	false	false	true	"true"
true	false	false	false	"true"
false	true	true	true	"true"
false	true	true	false	"true"
false	true	false	true	"true"
false	true	false	false	"true"
false	false	true	true	"false"
false	false	true	false	"true"
false	false	false	true	"true"
false	false	false	false	"true"

b. boolean\truthlists()
The truthtable() function is sufficient for most purposes unless the augmented Boolean expression contains non-Boolean variables or the truth table is for multiple Boolean expressions. This function fills that void. It accepts a list of strings of Boolean expressions and a list of Boolean variable names and builds the appropriate truth tables. Most frequently, the second argument

will use the listvars() function to generate the Boolean variable names as shown in the following examples:

```
boolean\truthlists({"a∧b","a∨b","a⊙b","a↓b","a↑b","a⇒b","a⇔b"},
boolean\listvars("a b"))
```

"a"	"b"	"a∧b"	"a∨b"	"a⊙b"	"a↓b"	"a↑b"	"a⇒b"	"a⇔b"
true	true	"true"	"true"	"false"	"false"	"false"	"true"	"true"
true	false	"false"	"true"	"true"	"false"	"true"	"false"	"false"
false	true	"false"	"true"	"true"	"false"	"true"	"true"	"false"
false	false	"false"	"false"	"false"	"true"	"true"	"true"	"true"

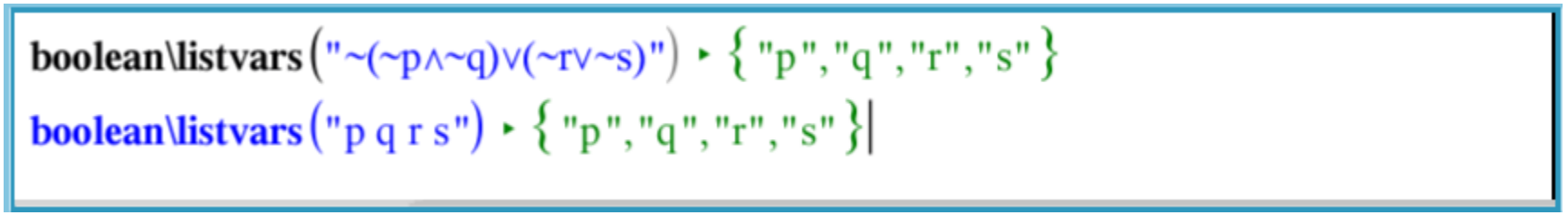
```
boolean\truthlists({"~p⇒~q","p∨~q","~(p∨~q)","(~p⇒~q)∧~(p∨~q)"},
boolean\listvars("~p⇒~q"))
```

"p"	"q"	"~p⇒~q"	"p∨~q"	"~(p∨~q)"	"(~p⇒~q)∧~(p∨~q)"
true	true	"true"	"true"	"false"	"false"
true	false	"true"	"true"	"false"	"false"
false	true	"false"	"false"	"true"	"false"
false	false	"true"	"true"	"false"	"false"

c. boolean\listvars()

This function takes a Boolean expression or a string containing Boolean variable names and outputs a list of strings of the

Boolean variable names as shown in this example:



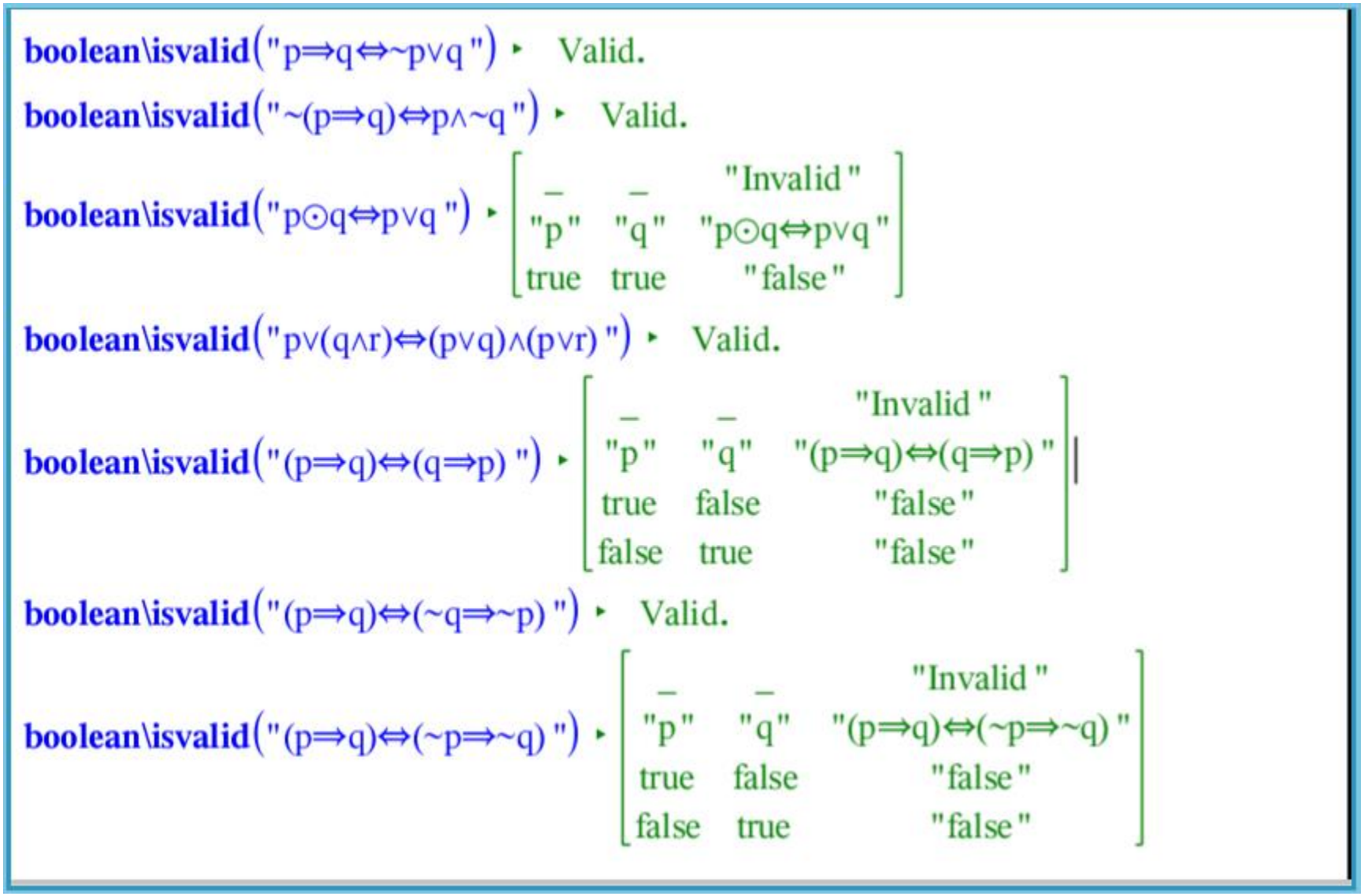
7. Proof Functions

These functions provide proofs for Boolean expressions.

a. IsValid()

This function takes a Boolean expression, extracts the Boolean variable names and builds a truth table for the variables and the expression. Next, it examines the table to determine the validity of the Boolean expression. If found to be true (all true values under the expression) it announces so without displaying the table. If any entries are found to be false, those rows only are

displayed showing the variable true/false entries that made the expression false.



b. Prove()

The prove() function takes two arguments. The first is a list of Boolean expressions that are premises assumed to be true. The

second is a Boolean expression the proof of which, with the given premises, are to be shown as valid or invalid. Again, a truth table is constructed columns for all the variables, the premise equation and the Boolean expression to be proven. If the table has all true entries under the expression to be proven, the proof is announced as valid without displaying the underlying table. If any

false entries are found, the proof is announced as invalid and the rows showing the combination/s that proved it invalid.

```
boolean\prove({"b","p⇒~b","~p⇒s"},"~s")
```

_	_	_	_	_	_	_	"Invalid"
"b"	"p"	"s"	"b"	"p⇒~b"	"~p⇒s"	"~s"	"((b)∧(p⇒~b)∧(~p⇒s))⇒
true	false	true	"true"	"true"	"true"	"false"	"false"

```
boolean\prove({"s∨i","s⇒(e∧t)","~e∨~t"},"i") ▸ Valid.
boolean\prove({"m⇒o","o⇒~p","o∨p"},"m⇒p")
```

_	_	_	_	_	_	_	"Invalid"
"m"	"o"	"p"	"m⇒o"	"o⇒~p"	"o∨p"	"m⇒p"	"((m⇒o)∧(o⇒~p)∧(o∨p)
true	true	false	"true"	"true"	"true"	"false"	"false"

```
boolean\prove({"y∨~p","~p⇒~n","n"},"y") ▸ Valid.
```

Given: "d⇒~w" Ducks don't waltz.

"o⇒w" Officers waltz.

"p⇒d" Poultry are ducks

Prove: "p⇒~w" Poultry don't waltz.

```
boolean\prove({"d⇒~w","o⇒w","p⇒d"},"p⇒~w") ▸ Valid.
```

8. Helper Function Listing

These functions help implement the user functions above and not intended for general usage so they are not documented further.

a. Boolean\ckvalid()
b. boolean\newfunction()
c. boolean\tofrombool()