On the Inexistence of a Unique Existential Binary Operator

João F. Ferreira
joao@joaoff.com

August 19, 2009

Abstract

We prove that there is not any boolean binary operator that can be quantified over an arbitrary set of values to express that exactly one of them is true. Our proof is a Haskell program that verifies this fact for all sixteen possible boolean binary operators.

1 Introduction

Boolean inequality, usually denoted by $\not\equiv$ and sometimes called exclusive-or, can be used to express that exactly one of two values is true. However, we can not use it to express that exactly one of three values is true (e.g. $\mathsf{true} \not\equiv \mathsf{true} \not\equiv \mathsf{true}$ is true and all of the operands are true). In this note, we prove that there is not any binary operator that can be quantified over an arbitrary set of values to express that exactly one of them is true. Our proof is a Haskell program that, for all binary boolean operators $\oplus$, shows that it is impossible to evaluate $(a \oplus b) \oplus c$ or $a \oplus (b \oplus c)$ to true exactly when one of a, b, and c is true and to false otherwise.

2 Boolean Binary Operators

We start by defining some useful datatypes. A binary operator $\oplus$ has type *BinOp*; it is represented as a list of pairs $(Input, Output)$, where *Input* is a pair of booleans (a, b) and the *Output* is the result of $a \oplus b$.

$$\begin{aligned}
&\textbf{type } \mathit{Input} &&= (\mathit{Bool}, \mathit{Bool}) \\
&\textbf{type } \mathit{Output} &&= \mathit{Bool} \\
&\textbf{type } \mathit{BinOp} &&= [(\mathit{Input}, \mathit{Output})]
\end{aligned}$$

The value *boolvars* is defined for convenience.

$$\mathit{boolvars} = [\mathit{True}, \mathit{False}]$$

The function *inputs* lists all four possible inputs.

$$\begin{aligned}
&\mathit{inputs} :: [\mathit{Input}] \\
&\mathit{inputs} = [(a, b) \mid a \leftarrow \mathit{boolvars}, \\
&\qquad\qquad\qquad\quad b \leftarrow \mathit{boolvars}]
\end{aligned}$$

The function *outputs* lists all sixteen possible outputs.

$$\begin{aligned}
&\mathit{outputs} :: [[\mathit{Output}]] \\
&\mathit{outputs} = [[a, b, c, d] \mid a \leftarrow \mathit{boolvars}, \\
&\qquad\qquad\qquad\qquad\quad b \leftarrow \mathit{boolvars}, \\
&\qquad\qquad\qquad\qquad\quad c \leftarrow \mathit{boolvars}, \\
&\qquad\qquad\qquad\qquad\quad d \leftarrow \mathit{boolvars}]
\end{aligned}$$

The first element returned by *outputs* corresponds to the binary operator *constant true*, as the following snippet shows:

$\gg$ head outputs
$[\mathit{True}, \mathit{True}, \mathit{True}, \mathit{True}]$

Finally, the function *operators* returns the list of all the sixteen boolean binary operators.

$$\begin{aligned}
&\mathit{operators} :: [\mathit{BinOp}] \\
&\mathit{operators} = \mathit{map}\ (\mathit{zip}\ \mathit{inputs})\ \mathit{outputs}
\end{aligned}$$

The first element returned by *operators* is the binary operator *constant true*:

$\gg$ head operators
$[((\mathit{True}, \mathit{True}), \mathit{True}), ((\mathit{True}, \mathit{False}), \mathit{True}), ((\mathit{False}, \mathit{True}), \mathit{True}), ((\mathit{False}, \mathit{False}), \mathit{True})]$

3 Unique Existential Operator

Now, recall that our goal is to check the value of the two following expressions, for all booleans a, b, and c, and for all boolean binary operators $\oplus$:

$$(a \oplus b) \oplus c \quad \text{and} \quad a \oplus (b \oplus c) \quad .$$

First, we generate all possible inputs for the above expressions.

$$
\begin{aligned}
&tinps :: [(Bool, Bool, Bool)]\\
&tinps = [(a, b, c) \mid a \leftarrow boolvars,\\
&\qquad\qquad\qquad\quad\; b \leftarrow boolvars,\\
&\qquad\qquad\qquad\quad\; c \leftarrow boolvars]
\end{aligned}
$$

Second, we define two functions, *checkl* and *checkr*, that given a list of inputs and a binary operator, evaluates each of the inputs using that operator. *checkl* associates to the left and *checkr* associates to the right.

$$
\begin{aligned}
&checkl && :: [(Bool, Bool, Bool)] \rightarrow BinOp \rightarrow [Output]\\
&checkl\ []\quad\ \_ && = []\\
&checkl\ ((a, b, c) : xs)\ op && = checkop\ op\ ((checkop\ op\ (a, b)), c) : checkl\ xs\ op
\end{aligned}
$$

$$
\begin{aligned}
&checkop && :: BinOp \rightarrow Input \rightarrow Bool\\
&checkop\ ((e, r) : es)\ p && \mid e == p \quad\ = r\\
& && \mid otherwise = checkop\ es\ p
\end{aligned}
$$

$$
\begin{aligned}
&checkr && :: [(Bool, Bool, Bool)] \rightarrow BinOp \rightarrow [Output]\\
&checkr\ []\quad\ \_ && = []\\
&checkr\ ((a, b, c) : xs)\ op && = checkop\ op\ (a, (checkop\ op\ (b, c))) : checkr\ xs\ op
\end{aligned}
$$

We can now evaluate all the possible outputs for the expression $(a \oplus b) \oplus c$ by mapping the function *checkl tinps* over the list *operators*. Symmetrically, we can evaluate all the possible outputs for the expression $a \oplus (b \oplus c)$ by mapping the function *checkr tinps* over the list *operators*.

$$
\begin{aligned}
&expl :: [[Output]]\\
&expl = map\ (checkl\ tinps)\ operators
\end{aligned}
$$

$$
\begin{aligned}
&expr :: [[Output]]\\
&expr = map\ (checkr\ tinps)\ operators
\end{aligned}
$$

We now want to filter all the operators that have exactly three true output entries. We start by defining the function *fthree* that tests if a given list of outputs has exactly three true elements.

$$
\begin{aligned}
&fthree :: [Output] \rightarrow Bool\\
&fthree = (== 3) \circ length \circ filter\ (== True)
\end{aligned}
$$

We then map *fthree* over *expl* and *expr*.

$$tmp1 = map\ fthree\ expl$$
$$tmp2 = map\ fthree\ expr$$

Now, using the next function, *flist*, we can combine the lists of booleans constructed by *tmp1* and *tmp2* with the list *operators* to filter the operators we are interested in.

$$flist :: [Bool] \to [BinOp] \to [BinOp]$$
$$flist = (map\ snd \circ) \circ (filter\ fst \circ) \circ zip$$

And so, the only operators that are of interest are:

$$opl :: [BinOp]$$
$$opl = flist\ tmp1\ \ operators$$

$$opr :: [BinOp]$$
$$opr = flist\ tmp2\ \ operators$$

For all the operators $\oplus$ in *opl*, we know that $(a \oplus b) \oplus c$ returns exactly three true values. Symmetrically, we also know that for all the operators $\oplus$ in *opr*, $a \oplus (b \oplus c)$ returns exactly three true values.

It remains to check if these three true values correspond to exactly when one of the arguments is true. We create a list with the three input combinations where exactly one is true.

$$oneistrue = [(True, False, False), (False, True, False), (False, False, True)]$$

Finally, using the operators in *opl* and *opr*, we determine if these inputs evaluate to true.

$$resultl :: Bool$$
$$resultl = and\ \$\ map\ and\ \$\ map\ (checkl\ \ oneistrue)\ opl$$

$$resultr :: Bool$$
$$resultr = and\ \$\ map\ and\ \$\ map\ (checkr\ \ oneistrue)\ opr$$

As the following snippet shows, both *resultl* and *resultr* evaluate to false. We can thus conclude that there is not any boolean binary operator that can be quantified over an arbitrary set of values to express that exactly one of them is true.

≫ resultl
False
≫ resultr
False