Ternary Exclusive *Or*

Francis Jeffry Pelletier, *Department of Philosophy, Simon Fraser University, Burnaby B.C., Canada V5A 1S6. email: jeffpell@sfu.ca*

Andrew Hartline, *Department of Philosophy, Simon Fraser University, Burnaby B.C., Canada V5A 1S6. email: ahartlin@sfu.ca*

Abstract

Ternary exclusive *or* is the (two valued) truth function that is true just in case exactly one of its three arguments is true. This is an interesting truth function, not definable in terms of the binary exclusive *or* alone, although the binary case is definable in terms of the ternary case. This article investigates the types of truth functions that can be defined by ternary exclusive *or*, and relates these findings to the seminal work of Emil Post.

Keywords: truth functions, Emil Post, functional completeness, *or*.

1 Introduction

Binary inclusive *or* is the (two-valued) truth function described in the table

φ_1	φ_2	$(\varphi_1 \vee \varphi_2)$
T	T	T
T	F	T
F	T	T
F	F	F

That is, a sentence whose main connective is $\vee$ is true just in case at least one of its two arguments is true, and is false otherwise. Elementary logic textbooks point out that $\vee$ is associative, and so internal parentheses of complex formulas made up entirely of $\vee$'s can be dropped. It is also often pointed out that $\vee$ is commutative, so that all orders of stating the arguments are equivalent. Together these two facts are sometimes used to justify a "prenex" form of $\vee$ that is seen as having variable adicity:

$$\vee(\varphi_1, \varphi_2, \ldots, \varphi_n)$$

which says "at least one of $\varphi_1, \varphi_2, \ldots, \varphi_n$ is true".

Binary exclusive *or* is the (two-valued) truth function described in the table

φ_1	φ_2	$(\varphi_1 \oplus \varphi_2)$
T	T	F
T	F	T
F	T	T
F	F	F

That is, a sentence whose main connective is $\oplus$ is true just in case exactly one of its two arguments is true, and is false otherwise. In particular, it is false when both of its arguments are true. $\oplus$ is associative, so internal parentheses of complex formulas made up entirely of $\oplus$'s can be dropped. It is also true that $\oplus$ is commutative, so that all orders of stating the arguments are equivalent. Together these two facts can be used to justify a "prenex" form of $\oplus$ that can be seen as having variable adicity:

$\oplus(\varphi_1, \varphi_2, \ldots, \varphi_n)$

However, this formula is *not* correctly understood as saying "exactly one of $\varphi_1, \varphi_2, \ldots, \varphi_n$ is true", as can be seen from the simple case of n=3.

φ_1	φ_2	φ_3	$((\varphi_1 \oplus \varphi_2) \oplus \varphi_3)$
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	T
F	F	F	F

Here we see that $\oplus(\varphi_1, \varphi_2, \varphi_3)$ is true not only when exactly one of $\varphi_1, \varphi_2, \varphi_3$ is true but also when all three of them are true. So the argument form that was used in the inclusive *or* case to project "is true if at least one of the arguments is true" from the binary $\vee$ to the prenex variable adicity $\vee$ is not in general a valid form of argumentation: it makes use of some other features of the inclusive *or* that are not present with exclusive *or*.

The reason that $\vee$ does intuitively allow for a variable-adicity form but $\oplus$ does not is related to the fact that there is a "natural" definition of $\vee^n(\varphi_1, \varphi_2, \ldots, \varphi_n)$, for each n, but the same "natural" definition of $\oplus^n(\varphi_1, \varphi_2, \ldots, \varphi_n)$ does not yield the meaning that it is true just in case exactly one of the arguments is true. For the former we have[1]

$\vee^2(\varphi_1, \varphi_2) \;=\; (\varphi_1 \vee \varphi_2)$

$\vee^n(\varphi_1, \varphi_2, \ldots, \varphi_n) \;=_{df}\; (\vee^{n-1}(\varphi_1, \varphi_2, \ldots, \varphi_{n-1}) \;\vee\; \varphi_n)$

And with this general definition of $\vee^n$ for all n, we might thereby justify the variable-adicity version, which we symbolize just as $\vee$, and say it means that "at least one of the following is true". But as we have just seen, when this type of inductive clause is used for $\oplus^n$, the resultant formula does not say "exactly one of the following is true". In the case of $n = 3$ it asserts that either exactly one of the following is true or else they all are true. More generally, $\oplus^n(\varphi_1, \varphi_2, \ldots, \varphi_n)$ is true just in case an odd number of the arguments $\varphi_1, \varphi_2, \ldots, \varphi_n$ are true. This is called the property of being an *odd counting function* of adicity n, in the terminology of [4]. For $n \geq 2$, all of the $\oplus^n$ are odd counting functions. It is shown in [4] that any composition of odd counting functions is itself an odd counting function.[2] Since an n-ary connective ($n \geq 3$) that is true just in case exactly one of its arguments is true would not be an odd counting function (for example if all three arguments were true then the function would be false), it follows that no such connective can be defined by $\oplus$.[3]

2 Meanings of *Or*

It is standard in logic textbooks to distinguish inclusive from exclusive *or* – to distinguish the binary connectives $\vee$ and $\oplus$. And there is a cottage industry in the philosophy of language and formal semantics literature on whether there is or isn't an

[1] Alternatively, we could start with $\vee^1(\varphi) \;=\; \varphi$.

[2] With a caveat concerning "dummy arguments". For details see [4].

[3] Nor by any of the particular $\oplus^n$'s alone, since they are all counting functions, nor by any combination of them.

exclusive *or* in natural language. But another cleavage, more relevant to the present paper, is a distinction between meanings of the phrase 'exclusive *or*'. Besides the issue of adicity of *or*, there is also the issue of whether we think of exclusive *or* as meaning 'exactly one of the following' or as meaning 'being connected by $\oplus$'. If we think in the former way, we will be unhappy with iterations of $\oplus$, for the reasons we have just surveyed. Indeed, iterating $\oplus$ amounts to addition modulo 2, and while there are plenty of reasons to like this operation, it doesn't express the 'exactly one of the following' sense of exclusive *or*. For the 'exactly one of the following n items' sense of exclusive *or* we will use the symbol $\veebar^n$; and we will use the symbol $\veebar$ for the variable adicity version of this sense; we thereby distinguish $\veebar$ from both $\oplus$ and $\vee$.

An interesting confusion arises because the two notions of exclusive *or* agree for $n = 2$: $\oplus^2(\varphi_1, \varphi_2)$ and $\veebar^2(\varphi_1, \varphi_2)$ express the same truth table. One might want to say that the two notions of exclusive *or*, $\oplus$ and $\veebar$, are "extensionally equivalent" in the binary case, but are "intensionally distinct" because their extensions in other adicities are different. The logic textbooks concentrate primarily on the binary case, and so they do not distinguish these two connectives. In the case of natural language, it seems pretty clear that when people employ an exclusive *or*, as in "You can have either the steak or the chicken or the fish dinner", they are using the 'exactly one' sense of *or* and not the 'addition modulo 2' sense.

But there are also uses for iterations of $\oplus$ (that thereby define $\oplus^n$). A natural use is in establishing the parity of an n-bit message, so that this information can be sent along with the message and the receiver can determine whether there has been some error in transmission by comparing the parity of the message with the extra sent bit. It also can find use in fast adders, where a simple $\oplus^n$ gate can determine the bit value of adding n-(binary)-numbers. (See [2, Chap. 5] for details).

Our interest is in the formal properties of the 'exactly one' sense of exclusive *or*, since that topic has not been addressed by the logic textbooks (nor by the formal semantic descriptions of natural language). We will be calling this connective the "real" variable-adicity exclusive *or*, meaning thereby that it is the one that is relevant to formal accounts of natural language. We think that $\oplus^n$ might better be called "the odd counting function of adicity n", and that iterations of $\oplus$ should be called "addition modulo 2" rather than "exclusive disjunction".

3 Defining $\veebar$

Of course, with a functionally complete set of connectives like $\{\vee, \neg\}$, $\{\rightarrow, \bot\}$, $\{\uparrow\}$ (NAND), or $\{\downarrow\}$ (NOR), we can define any connective. So, we could use such a set of connectives to define all the $\veebar^n$, for any n. And if one allows variables in the syntax, one can even give a formula that expresses the general claim:

$$\veebar^n(\varphi_1, \varphi_2, \ldots, \varphi_n) \;=_{df}\; (\vee^n(\varphi_1, \varphi_2, \ldots, \varphi_{n-1}) \wedge \bigwedge_{i<j\leq n} \neg(\varphi_i \wedge \varphi_j)$$

But that is not the method we want to pursue here. Intuitively, there should be some way to use some one exclusive *or* to define all the other exclusive *or*s without recourse to other connectives. That is, there should be some analogue to the method we employed to define all the particular inclusive $\vee^n$s, and thereby employed in accounting for the variable-adicity inclusive $\vee$.

The key is to start with a ternary exclusive *or* rather than the usual binary exclusive *or*. We designate this connective as $\underline{\vee}^3$; primitive formulas with this as main connective take the form $\underline{\vee}^3(\varphi_1, \varphi_2, \varphi_3)$. Using only this connective we can define $\bot$ (the constant **false**) as follows:

$$\bot =_{df} \underline{\vee}^3(\varphi_1, \varphi_1, \varphi_1)$$

(If φ_1 is true, then there are three true arguments to $\underline{\vee}^3$, and hence it is false; if φ_1 is false, then there are zero true arguments to $\underline{\vee}^3$, and again it is false.)

The usual binary exclusive *or*, which we above simply called $\oplus$ when we used it as a normal infix operator $(\varphi_1 \oplus \varphi_2)$ and which we identified with the prefix operator of adicity 2, $\oplus^2(\varphi_1, \varphi_2)$, would be called $\underline{\vee}^2$ ("exactly one of the two arguments is true") in this new notation. This connective can be defined using our new $\underline{\vee}^3$ as follows:

$$\underline{\vee}^2(\varphi_1, \varphi_2) =_{df} \underline{\vee}^3(\varphi_1, \varphi_2, \bot) \qquad [= \oplus^2(\varphi_1, \varphi_2)]$$

(If both φ_1 and φ_2 are true, then there are two true arguments to $\underline{\vee}^3$, and the formula is false; if they are both false, then there are zero true arguments to $\underline{\vee}^3$ and again the formula is false. It is true when and only when exactly one of φ_1 and φ_2 is true.)

We can also define a binary *and-not* connective that we will write as $[\wedge\neg]$. Here we employ it as an infix connective, but we will shortly define $[\wedge\neg]^n$ as a prefix connective; so, $[\wedge\neg]^2$ is ambiguously both an infix and a prefix binary operator:

$$(\varphi_1\ [\wedge\neg]\ \varphi_2) =_{df} \underline{\vee}^3(\varphi_1, \varphi_2, \varphi_2)$$

(Note first that this formula could never be true if φ_2 is true, since then there would be at least two true arguments. So φ_2 must be false in order for the formula to be true. Hence, if it is true, this must be on account of φ_1 being true. That is, φ_1 is true and φ_2 is false – which is the binary and-not connective.)

Now consider

$$\underline{\vee}^3([\varphi_1\ [\wedge\neg]\ \varphi_2], \varphi_3, \varphi_3)$$

[i.e., $\underline{\vee}^3(\underline{\vee}^3(\varphi_1, \varphi_2, \varphi_2), \varphi_3, \varphi_3)$, when unabbreviated]. This is identical to

$$((\varphi_1\ [\wedge\neg]\ \varphi_2)\ [\wedge\neg]\ \varphi_3)$$

And generally, we can easily define an extended n-ary *and-not* connective $[\wedge\neg]^n(\varphi_1, \varphi_2, \ldots \varphi_n)$ which is true if φ_1 is true and all of $\varphi_2 \ldots \varphi_n$ are false, and is false otherwise. The inductive step of this definition, for $n \geq 3$ is

$$[\wedge\neg]^n(\varphi_1, \varphi_2, \ldots \varphi_n) =_{df} ([\wedge\neg]^{n-1}(\varphi_1, \varphi_2, \ldots \varphi_{n-1})\ [\wedge\neg]\ \varphi_n)$$

Before embarking on the general definition of $\underline{\vee}^n$, let's warm up with a definition of $\underline{\vee}^4$, using just our $\underline{\vee}^3$ and the connectives we have already defined by using just $\underline{\vee}^3$. Consider the four formulas

$$[\wedge\neg]^4(\varphi_1, \varphi_2, \varphi_3, \varphi_4)$$
$$[\wedge\neg]^4(\varphi_2, \varphi_1, \varphi_3, \varphi_4)$$
$$[\wedge\neg]^4(\varphi_3, \varphi_1, \varphi_2, \varphi_4)$$
$$[\wedge\neg]^4(\varphi_4, \varphi_1, \varphi_2, \varphi_3)$$

Each of these formulas is true just in case its first argument is true and all the other arguments are false. Let's take the first three of these formulas and embed them under $\underline{\vee}^3$:

(1) $\underline{\vee}^3([\wedge\neg]^4(\varphi_1, \varphi_2, \varphi_3, \varphi_4), [\wedge\neg]^4(\varphi_2, \varphi_1, \varphi_3, \varphi_4), [\wedge\neg]^4(\varphi_3, \varphi_1, \varphi_2, \varphi_4))$

By the truth table for $\underline{\vee}^3$, this formula is true just in case exactly one of the embedded formulas is true – that is, just in case exactly one of the following occurs: (a) φ_1 is true and $\varphi_2, \varphi_3, \varphi_4$ are false, or (b) φ_2 is true and $\varphi_1, \varphi_3, \varphi_4$ are false, or (c) φ_3 is true and $\varphi_1, \varphi_2, \varphi_4$ are false. Finally, consider

(2) $\underline{\vee}^2(\underline{\vee}^3([\wedge\neg]^4(\varphi_1, \varphi_2, \varphi_3, \varphi_4), [\wedge\neg]^4(\varphi_2, \varphi_1, \varphi_3, \varphi_4), [\wedge\neg]^4(\varphi_3, \varphi_1, \varphi_2, \varphi_4)),$

$[\wedge\neg]^4(\varphi_3, \varphi_1, \varphi_2, \varphi_4))$

where we have used binary exclusive *or* (represented here as the prefix $\underline{\vee}^2$) to disjoin (1) with $[\wedge\neg]^4(\varphi_4, \varphi_1, \varphi_2, \varphi_3)$. Formula (2) is true just in case one of (a), (b), (c) or (φ_4 is true and $\varphi_1, \varphi_2, \varphi_3$ are false). But this means that formula (2) is true just in case exactly one of $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ are true and all the others are false. And this is precisely $\underline{\vee}^4$, defined entirely in terms of $\underline{\vee}^3$. In unabbreviated form it is

$$\begin{aligned}\underline{\vee}^4(\varphi_1, \varphi_2, \varphi_3, \varphi_4) \;=_{df}\; \underline{\vee}^3(\underline{\vee}^3(\underline{\vee}^3(\underline{\vee}^3(\underline{\vee}^3(\varphi_1, \varphi_2, \varphi_2), \varphi_3, \varphi_3), \varphi_4, \varphi_4),\\ \underline{\vee}^3(\underline{\vee}^3(\underline{\vee}^3(\varphi_2, \varphi_1, \varphi_1), \varphi_3, \varphi_3), \varphi_4, \varphi_4),\\ \underline{\vee}^3(\underline{\vee}^3(\underline{\vee}^3(\varphi_3, \varphi_1, \varphi_1), \varphi_2, \varphi_2), \varphi_4, \varphi_4)),\\ \underline{\vee}^3(\underline{\vee}^3(\underline{\vee}^3(\varphi_4, \varphi_1, \varphi_1), \varphi_2, \varphi_2), \varphi_3, \varphi_3),\\ \underline{\vee}^3(\varphi_1, \varphi_1, \varphi_1))\end{aligned}$$

It is pretty easy to see that this method can be generalized to yield $\underline{\vee}^n$, for any n. The idea is to inductively use $\underline{\vee}^{n-1}$, $\underline{\vee}^2$, and $[\wedge\neg]^n$ to define $\underline{\vee}^n$. We have already seen that $\underline{\vee}^2$ and $[\wedge\neg]^n$ (for any n) can be defined using only $\underline{\vee}^3$. We need only describe the way to generate $\underline{\vee}^n$ given that we have $\underline{\vee}^2$ and $[\wedge\neg]^n$, and by induction, $\underline{\vee}^{n-1}$. Recall that we have a way to say that exactly one of n subformulas are true while all the others are false, let's look at this list of formulas, each one of which says that its first argument is true and all its others are false:

$[\wedge\neg]^n(\varphi_1, \varphi_2, \ldots, \varphi_n)$
$[\wedge\neg]^n(\varphi_2, \varphi_1 \ldots, \varphi_n)$
$\ldots$
$\ldots$
$[\wedge\neg]^n(\varphi_{n-1}, \varphi_1, \ldots, \varphi_{n-2}, \varphi_n)$
$[\wedge\neg]^n(\varphi_n, \varphi_1, \ldots, \varphi_{n-1})$

Since we already have $\underline{\vee}^{n-1}$ by hypothesis, we can use the first $n-1$ formulas on this list and say

$$\begin{aligned}\underline{\vee}^{n-1}([\wedge\neg]^n(\varphi_1, \varphi_2, \ldots, \varphi_n), [\wedge\neg]^n(\varphi_2, \varphi_1, \varphi_3, \ldots, \varphi_n), \ldots,\\ [\wedge\neg]^n(\varphi_{n-1}, \varphi_1, \ldots, \varphi_{n-2}, \varphi_n)),\end{aligned}$$

asserting thereby that exactly one of the embedded formulas is true, i.e., that exactly one of $\varphi_1, \varphi_2, \ldots, \varphi_{n-1}$ is true while at the same time all the others are false, including φ_n. We can now complete the definition by doing a binary exclusive *or* of this formula with the last formula on the list:

$$\begin{aligned}\underline{\vee}^2(\underline{\vee}^{n-1}([\wedge\neg]^n(\varphi_1, \varphi_2, \ldots, \varphi_n), [\wedge\neg]^n(\varphi_2, \varphi_1, \varphi_3, \ldots, \varphi_n), \ldots,\\ [\wedge\neg]^n(\varphi_n, \varphi_1, \ldots, \varphi_{n-1}))\end{aligned}$$

which is true if and only if one or the other (but not both) of the two embedded formulas are true. That is, it is true iff either (i) exactly one of $\varphi_1, \varphi_2, \ldots, \varphi_{n-1}$ are true and all the others (including φ_n) are false, or (ii) φ_n is true and all of $\varphi_1, \varphi_2, \ldots, \varphi_{n-1}$ are false.

But that is precisely $\underline{\vee}^n(\varphi_1, \varphi_2, \ldots, \varphi_n)$: the formula that says that exactly one of the n disjuncts is true. The fact that $\underline{\vee}^n$ can algorithmically be defined from $\underline{\vee}^3$ without using any other connectives, gives a justification for using $\underline{\vee}$ (without a superscript) as a variable adicity exclusive *or*.

4 A Small Puzzle

It needs of course to be remembered that $\underline{\vee}$ is not generated using the ordinary binary exclusive *or* connective, $\oplus^2$, as a basis, but rather the ternary exclusive *or*, $\underline{\vee}^3$. Even

when this variable adicity connective is used with two arguments, it is not the case that such a use of the connective imports $\oplus^2$. Instead it is relying on $\veebar^2$, which in turn relies on $\veebar^3$.

But, one might ask, why is $\veebar^3$ the required primitive, and not $\veebar^2$? The construction primarily employs $\bot$ and $[\wedge\neg]$. $\bot$ is definable from $\veebar^2$ (indeed, from $\oplus^2$), and the most salient surface feature of $[\wedge\neg]$ which is clearly not a feature of $\veebar^2$ is that it is non-commutative. Yet $\veebar^3$ *is* commutative between any of its positions. Given all this, it is a puzzle how it is that $\veebar^3$ can define a non-commutative connective but $\veebar^2$ cannot and why, therefore, $\veebar^3$ can't be defined from $\veebar^2$.

The answer is that, extensionally, $\veebar^2$ is a counting function, like $\oplus^2$, and so all iterations are also (equivalent to) counting functions and can only define functions that have the same number of **T**'s and **F**'s in their truth tables (modulo issues concerning dummy variables). But $\veebar^3$ is not a counting function: iterating $\veebar^3$ does not "keep track of" how many arguments are true/false. So, while it may appear that the important difference is that $[\wedge\neg]$ is not commutative, unlike $\oplus^2$, in reality the important difference is "counting" (or rather, being *non*-counting): $\veebar^2$ is a counting function and $\veebar^3$ isn't.

5 The Power of *Or* – Ternary Exclusive *Or*, that is

Ternary exclusive *or* is an interesting connective. In the terminology of [4] it is an **F**-*preserving* connective because when its arguments are all **F** then the value it assumes is **F**. In this, it is similar to $\wedge$ and $\vee$, but different than $\rightarrow$, $\leftrightarrow$, and $\neg$. Being **F**-preserving is a property that is preserved by any composition of **F**-preserving connectives, and therefore any formula made up only of $\veebar^3$ will have a **F** in its last row (the row where all input values are **F**). So, unlike $\uparrow$ and $\downarrow$, it is not by itself a truth functionally complete connective.

However, it is **F**-*preservingly complete*: any connective that has a **F** in its last row can be defined using only $\veebar^3$; that is, all **F**-preserving connectives can be defined by $\veebar^3$ alone. This can be shown in a way similar to that used in the full propositional logic to show how to construct a formula in disjunctive normal form for any arbitrary truth table.

Consider, then, any truth table that has an **F** in its last row. There are two cases. In case one, *all* rows have the value **F**, and the truth table expresses a contradiction. We have already seen how to define the constant false $\bot$ as $\veebar^3(\varphi, \varphi, \varphi)$, so therefore this **F**-preserving truth function can be described by $\veebar^3$. The second case is where there is at least one row where the value of the truth function to be defined is **T**, as well as the **F** in the last row.

Here are a few preliminary definitions to be used in the second case. We first define binary *and*, ambiguously calling it the infix connective $\wedge$ and the prefix connective $\wedge^2$:

$(\varphi_1 \wedge \varphi_2) \ =_{df} \ [\wedge\neg]^2(\varphi_1, [\wedge\neg]^2(\varphi_1, \varphi_2))$

(Note that, since $[\wedge\neg]^2$ says that the first argument is true and the second argument is false, the first argument of the outer $[\wedge\neg]^2$, φ_1, must be true while $[\wedge\neg]^2(\varphi_1, \varphi_2)$ must be false. But if this latter is false, then either φ_1 must be false or φ_2 must be true. However, we already said that φ_1 is true, so therefore it must be that φ_2 is true. Thus both of φ_1 and φ_2 must be true in order for the right side of the definition to

be true. But this is the definition of $\wedge$.)

This is the base case of an inductive definition of $\wedge^n$, and here is the inductive step for $n \geq 3$:

$\wedge^n(\varphi_1, \ldots, \varphi_n) \;=\; (\wedge^{n-1}(\varphi_1, \ldots, \varphi_{n-1}) \wedge \varphi_n)$

Finally, with judicious use of $\wedge^n$ and $[\wedge\neg]^m$ we can express the concept "the following n statements are true, while the next m statements are false (for $n \geq 1,\ m \geq 0$)", as follows:

if $m = 0$ then $\wedge^n(\varphi_1, \ldots, \varphi_n)$, otherwise

$$[\wedge\neg]^{m+1}(\ \wedge^n(\varphi_1, \ldots, \varphi_n), \varphi_{n+1}, \ldots, \varphi_{n+m})\ )$$

(Note that if $m = 0$, then we use just $\wedge^n$. Note also that, by these definitions, at least one of the $\varphi_i, 1 \leq i \leq n$, must be true.)

With this, we are able to write a formula that will describe any input vector of an $(n+m)$-ary truth function (so long as it has at least one **T** input value). That is, for each row of the truth table for an arbitrary formula having $(n+m)$ different atomic variables, we can write an expression that "describes" the input values of any row of its truth table (except for the last row, which contains only **F**'s). If the formula contains, for example, five different atomic variables $(\varphi_1, \ldots, \varphi_5)$, then the truth table contains 32 rows. In the standard ordering of the rows, where the input vector for the first row contains all **T**'s, the eighteenth row of such a truth table will have φ_1 and φ_5 be **F** and the other three variables be **T**. The formula that describes this input vector is:

$[\wedge\neg]^4(\ \wedge^2(\varphi_2, \varphi_3, \varphi_4), \varphi_1, \varphi_5)$

(That is, φ_1 and φ_5 are **F** while $\varphi_2,\ \varphi_3$ and φ_4 are **T**).

Clearly, any row of an arbitrary truth table can be described in this way (except for the last row, which has no **T** atomic variables). And equally obvious is the fact that any two rows of a truth table are distinct in their assignments of **T** and **F** to atomic variables, so that for any two formulas that represent different rows of the truth table, at most one could take the value **T**.

So now back to our "case two", where there is at least one **T** value of the formula in addition to the **F** value when all input variables are **F**. As with the standard construction of a disjunctive normal form, we consider all the rows where the value of the formula is **T** (knowing that there is at least one such row). For each one of these rows we construct the expression that "describes" this row, using the method just given. If there were only one such row of the truth table because the initial formula has only one **T** value, then the expression constructed this way will have the same truth table. If there is more than one **T** row of the truth table (say there are k **T** rows), then we use $\veebar^k$ to exclusively disjoin them. Since no pair of these k expressions can simultaneously be **T**, it follows that this exclusive k-disjunction has the same truth table as the original formula.

But in any of these cases, all the connectives were defined using only $\veebar^3$. So, $\veebar^3$ is **F**-*preservingly complete*: any formula can be expressed using only $\veebar^3$, so long as it has the value **F** when all the input variables have the value **F**. The set of connectives containing $\veebar^3$ and any connective that is not **F**-preserving, for example the constant $\top$, is functionally complete. (This follows from [5], as explained in [4].)

6 A Final Historical Remark

As with most topics that concern properties of two-valued truth tables, the present result is closely connected with material in [5]. Post analyzed classes of truth-functions and discussed their generators. In the terminology of the current paper, he described classes of truth-functions all of whose members manifested certain properties, such as returning a **F** value when all the input values are **F**; or returning a **T** value when all the input values are **T**; or being self-dual; or being monotonic. One of the results of Post's work is recounted in [4]: the necessary and sufficient conditions for generating the set of all truth functions using only a subset of the functions.

The result of the present paper, that $\veebar^3$ is **F**-preservingly complete, can be recast in Post's terms like this: $\veebar^3$ is a generator of (Post's) category C_3. Post generated this category by considering all truth functions which, when all their variables are identified as being the same variable, say p, either yield the identity function (i.e., the same truth table as p) or else yield $\bot$ (constant false). If we are already given Post's proof of this, then the present result amounts to saying that Post's category C_3 contains all and only truth functions with **F** in their last line, and that C_3 is generated by ternary exclusive *or*, $\veebar^3$. Category C_3 is also generated by $\{\oplus, \wedge, \bot\}$ and by $\{\vee, [\wedge\neg]^2\}$, as remarked but not proved in [3]. Given these claims, the present paper can be seen as showing that $\oplus$, $\wedge$, $\bot$, $\vee$ and $[\wedge\neg]^2$ are definable by $\veebar^3$.

Post's insights concerning the abstract properties of truth functions are immensely interesting and his results should be better known. A nice place to start is with [6]. [1] is also a good source, although perhaps more advanced.

Acknowledgments.

Thanks to Alasdair Urquhart for discussions both about Post generally and about the topic of this paper. Thanks also to him for showing us a preprint of [6]. And thanks to Piotr Rudnicki for discussions concerning the usefulness of $\oplus^n$.

References

[1] S. Gindikin. *Algebraic Logic*. Springer-Verlag, NY, 1985. (A translation of Gindikin (1972) *Algebra Logiki v Zadachakh* Nauka. R.H. Silverman, translator).

[2] I. Koren. *Computer Arithmetic Algorithms, 2nd Ed.* A.K. Peters, Natick, MA, 2002.

[3] R. Lyndon. Identities in two-valued calculi. *Transactions of the American Mathematical Society*, 71:457–465, 1951.

[4] F. J. Pelletier and N. Martin. Proving Post's functional completeness theorem. *Notre Dame Journal of Formal Logic*, 31:462–475, 1990.

[5] E. Post. *The Two-Valued Iterative Systems of Mathematical Logic*. Princeton University Press, Princeton, 1941.

[6] A. Urquhart. Emil Post. In D. Gabbay and J. Woods, editors, *Handbook of the History of Logic*, volume 5: *Logic from Russell to Church*. Elsevier, London, forthcoming.

References

[1] Gindikin, S., *Algebraic Logic*, Springer-Verlag, NY, 1985. (A translation of Gindikin (1972) *Algebra Logiki v Zadachakh* Nauka. R.H. Silverman, translator).

[2] Koren, I., *Computer Arithmetic Algorithms, 2nd Ed.*, A.K. Peters, Natick, MA, 2002.

[3] Lyndon, R., 'Identities in two-valued calculi', *Transactions of the American Mathematical Society*, 71 (1951), 457–465.

[4] Pelletier, F. J., and N. Martin, 'Proving Post's functional completeness theorem', *Notre Dame Journal of Formal Logic*, 31 (1990), 462–475.

[5] Post, E., *The Two-Valued Iterative Systems of Mathematical Logic*, Princeton University Press, Princeton, 1941.

[6] Urquhart, A., 'Emil Post', in D. Gabbay, and J. Woods, (eds.), *Handbook of the History of Logic*, vol. 5: *Logic from Russell to Church*, Elsevier, London, forthcoming.

Received 18 March 2007