Designing Weave Structures Using Boolean Operations, Part 1

Introduction

Basic weave structures — interlacement patterns — can be described in many ways, but they all come down to representing the crossings of warp and weft threads. One or the other is on top. This is represented in drawdowns by a grid in which the cells represent the intersections and (usually) a cell is black if the warp thread is on top but white if the weft thread is on top. See Figure 1.

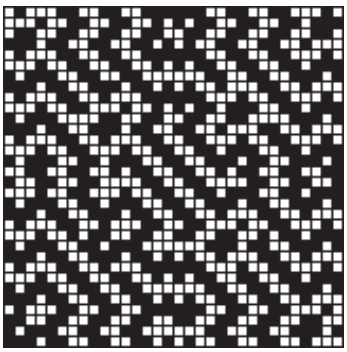


Figure 1. Drawdown

From a computational point of view, 1s and 0s instead of colors are the natural representation of interlacement patterns. See Figure 2, which corresponds to the drawdown of Figure 1. A 1 corresponds to black for the warp thread on top and a 0 corresponds to white for the weft thread on top.

```
01000110011001111110011001100010
00010011001100101011001100110111
01100110011001100110011001100110
11101100110011010100110011001000
01000110011001111110011001100010
00010011001100101011001100110111
10111001100110000001100110011101
11101100111011010100100011001000
01000110010001111110001001100010
00010011000100101011011100110111
10111000011001100110011000011101
11101100111011001100100011001000
01000110010001100110001001100010
00010011000100110011011100110111
10111001101110011001110110011101
00110011001011001100101100110011
10011001100001100110000110011001
00010011000100110011011100110111
10111001101110011001110110011101
11101100111011001100100011001000
01000110010001111110001001100010
00010011001100101011001100110111
10111001100110000001100110011101
11101100111011010100100011001000
01000110011001111110001001100010
00010011001100101011001100110111
10111001100110000001100110011101
11101100111011010100100011001000
01000110011001111110001001100010
11001100110011001100110011001100
10111001100110000001100110011101
11101100111011010100100011001000
```

Figure 2. Binary Interlacement Pattern

This binary representation suggests the use of Boolean operations for weave design.

Boolean Operations

George Boole (see the side-bar on the next page) invented the mathematical system named after him to describe logical operations on *truth values* — true or false [1].

Although Boole was motivated by logic, his system applies equally well to contexts in which there are two mutually exclusive values, such as "on" and "off". In the case of weave structures, the values are "warp on top" and "weft on top", or alternatively, "it is true that the warp is on top" and "it is false that the warp is on top". (The choice of warp rather than weft is arbitrary, as is the choice of 1 and 0 to represent truth values.)

In Boolean algebra [2], variables, which are indicated by x, y, and so forth, can have only two values — 1 or 0. Boolean operations produce values depending on the values of the variables to which they are applied. The operations, however, are not the familiar arithmetic ones, but rather logical operations like *and* and *or*.

Boolean operations are described by *truth tables* that detail the results depending on the value of the variables to which the operators are applied. An example is *not*, also known as *complement* and *negation*, indicated by the symbol ~. This operation has the truth table

x	$\sim x$
1	0
0	1

not

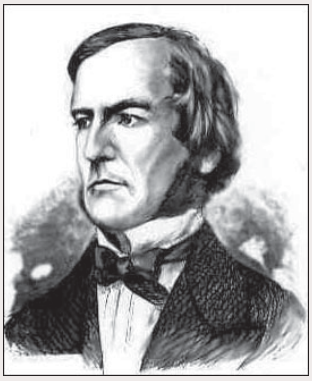
Other basic Boolean operations are *or* (+), also known as *disjunction*, and *and* (×), also known as *conjunction*. Their truth tables are

x	y	$x + y$
1	1	1
1	0	1
0	1	1
0	0	0

or

October 20, 2001; last revised June 20, 2002

George Boole



1815 - 1864

George Boole grew up in poverty in England. He early exhibited intellectual powers and showed a special aptitude for mathematics.

He was unable to pursue a formal education because he had to work to support his parents.

He undertook to learn mathematics on his own and soon began to do original work.

His work attracted the attention of prominent mathematicians and he began to publish in mathematical journals. He eventually was awarded a gold medal by the Royal Society.

He contributed to several areas of mathematics, but he is best known for his seminal work on the mathematics of logic.

This work is the foundation for modern computing and information technology.

Because of his need to work and lack of formal training in mathematics, he started late. Unfortunately, he died early.

He had hereditary lung disease. Caught in a drenching rain and late to a lecture, he continued without changing to dry clothes. He subsequently contracted pneumonia.

His wife, who thought the only way to cure a disease was to apply its cause, drenched him with cold water as he lay feverish in bed. He died shortly thereafter.

We can only speculate as to what Boole might have accomplished had he had the opportunity to get a formal education and had lived longer.

x	y	$x \times y$
1	1	1
1	0	0
0	1	0
0	0	0

and

Note that $\sim(x + y) = (\sim x \times \sim y)$ and $\sim(x \times y) = (\sim x + \sim y)$. These are known as De Morgan's Laws, named after the 19th century logician and mathematician Augustus De Morgan, who propounded them.

A particularly interesting Boolean operation is *exclusive or* ($\oplus$), which has the truth table

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

exclusive or

That is, $x \oplus y$ has the value 1 if exactly one of x and y is 1 — it excludes the case that both are. The + operations sometimes is called *inclusive or* to distinguish it from *exclusive or*.

Exclusive or has an interesting and important property:

$$(x \oplus y) \oplus y = x$$
$$(x \oplus y) \oplus x = y$$

That is, either x or y can be extracted from $(x \oplus y)$ by applying *exclusive or* with the other.

The complement of *exclusive or* is *equivalence* ($\leftrightarrow$), which has the value 1 only if x and y are the same.

x	y	$x \leftrightarrow y$
1	1	1
1	0	0
0	1	0
0	0	1

equivalence

Another important Boolean operation is *implication* ($\rightarrow$), which is based on *modus ponens*, one of the foundations of logical argument: If x implies y and x is true, then y is true.

x	y	$x \rightarrow y$
1	1	1
1	0	0
0	1	1
0	0	1

implication

All together there are 16 Boolean operations of two variables, corresponding to the 16 possible patterns of 0 and 1 for the four possible combinations of the variable values.

If we use the order of the values of x and y as given in the truth tables above, we can represent the 16 Boolean operations of two variables by the patterns of the values they produce.

For example, *or* and *and* have the value patterns

1110 *or*

1000 *and*

It is possible to use only a small *functionally complete* set of Boolean operations from which all 16 can be composed. Two functionally complete sets are {~, +} and {~, ×}. There are several other small functionally complete sets. In fact, there are single operations from which all others can be composed. See the sidebar **Sheffer Strokes**.

So far, we have identified Boolean operations by names and operator symbols. We can also identify them by the hexadecimal characters for their value patterns. For example, *or* has the hexadecimal identification e and *and* has the hexadecimal identification 8.

The complete list of 2-variable Boolean operations, with hexadecimal identifications and names and symbols for common operations is

0000	0		
0001	1		
0010	2		
0011	3		
0100	4		
0101	5		
0110	6	*exclusive or*	$x \oplus y$
0111	7		
1000	8	*and*	$x \times y$
1001	9	*equivalence*	$x \leftrightarrow y$
1010	a		
1011	b	*implication*	$x \rightarrow y$
1100	c		
1101	d		
1110	e	(*inclusive*) *or*	$x + y$
1111	f		

Sheffer Strokes

In a 1913 paper [1], Henry M. Sheffer showed that one of the 16 Boolean operations constitutes a functionally complete set in itself. This operation, called the Sheffer Stroke and designated by the symbol ↓, has the value pattern 0111 and the hexadecimal identification 7. This operation can be described as *not both x and y*.

It can be shown that

$$\sim x = x \downarrow y$$

and

$$x + y = (x \downarrow y) \downarrow (y \downarrow x)$$

Since it is well known that {~, +} constitutes a functionally complete set, then {↓} does also.

The second Sheffer Stoke, designated by the symbol ↑, has the value pattern 0001 and the hexadecimal identification 1. It also constitutes a functionally complete set.

While it may be interesting to know that only one operation is necessary to form all Boolean operations of two variables, using just one operation is complicated and unintuitive. These operations *are*, however, good for making up homework problems.

Reference

1. "A Set of Five Independent Postulates for Boolean Algebras, with Application to Logical Constants", *Transactions of the American Mathematical Society*, Vol. 14 (1913), pp. 481-488.

It is worth looking critically at this list. In the first place, eight of the operations are complements of the others. For example,

$$x \leftrightarrow y \;=\; \sim(x \oplus y)$$

Since complementation only changes 0s to 1s and vice versa, it can be done as a simple operation on the result.

In addition, the result of applying the operation 0, which has the value pattern 0000, is 0, regardless of the values of x and y. This operation therefore is of no use in design based on two values. The same is true of its complement, f.

Furthermore, the operation c, which has the value patterns 1100, always produces x regardless of the value of y. Similarly, the operation a, which has the value pattern 1010, always produces y regardless of the value of x. The complements of these operations produce $\sim x$ and $\sim y$, respectively.

Eliminating all the operations whose results do not depend on both x and y leaves 10 operations, of which five are complements of the other five.

The question then is what five operations to choose for design. It really doesn't matter, as long is none is the complement of another. We'll pick four that are considered most fundamental and add one other, *reverse implication*, $x \leftarrow y$:

$$x \times y$$
$$x + y$$
$$x \oplus y$$
$$x \rightarrow y$$
$$x \leftarrow y$$

Thus, a functionally complete *design* set is $\{\sim, \times, +, \oplus, \rightarrow, \leftarrow\}$.

Boolean Operations on Arrays

In the examples above, Boolean operations are applied to individual variables, such as x and y. The operations also can be applied to arrays of Boolean variables, denoted by X, Y, and so forth. When an operation is applied to two arrays, it is applied to all the values in corresponding positions of the two arrays to give a new array. Figure 3 shows an example.

X		Y		Z
1 0 1 0		1 1 0 0		0 1 1 0
0 1 0 1		0 1 1 0		0 0 1 1
1 0 1 0	$\oplus$	0 0 1 1	=	1 0 0 1
0 1 0 1		1 0 0 1		1 1 0 0
1 0 1 0		0 0 1 1		0 1 1 0

Figure 3. Boolean Operation on Arrays

Boolean operations on arrays are the basis for the methods of weave design described here. Boolean operations can be performed on interlacement patterns cast in the form of arrays of Boolean values, as shown in Figure 2. For example, Figure 3 shows the result of applying *exclusive or* to a small plain weave and a 2/2 twill to produce a new interlacement pattern (which is just a shifted 2/2 twill).

An Application — Diversified Weaves

Oelsner, whose well-known book was first published in 1915 [3], includes a chapter on what he calls *diversified weaves*. The chapter opens as follows:

> Very attractive patterns can be obtained by adding or removing risers from a ground weave. These alterations are made according to a previously selected motif.

Many of his examples classify as spot weaves, but he goes beyond that.

The most common ground weave is plain weave, although others can be used. Figure 4 shows a spot weave obtained by applying *or* to a plain weave and a repeated motif.

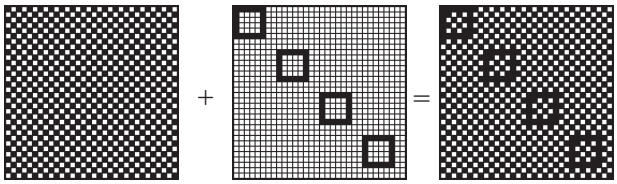


Figure 4. A Spot Weave

Other Boolean operations produce different patterns. The result of applying *exclusive or* is shown in Figure 5.

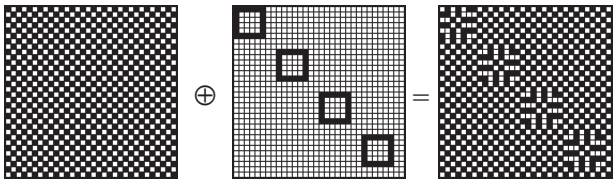


Figure 5. *Exclusive Or* Spot Weave

Other weaves can be used for the ground. Figure 6 shows a 2/2 twill in place of the plain weave in Figure 5.

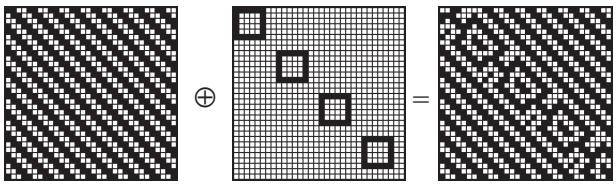


Figure 6. Twill-Based Spot Weave

Other Applications

The concept of combining figure and ground implies a fundamental distinction between the two. Boolean operations can, of course, be applied to any two weaves. The question is what kinds of weave and which operations produce good results. Figure 7 shows the result of combining a 2/2 twill with a basket weave.

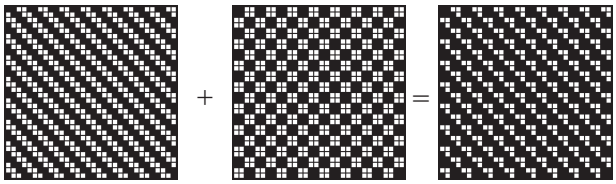


Figure 7. Combining Plain and Basket Weaves

Comments

Boolean operations provide a natural way to combine two (or more) interlacement patterns to form new ones. As in all matters of design, artistic sense and understanding of the tools used are essential.

There are two potential problems associated with using Boolean operations in weave design: long floats and the loom resources required.

In Boolean operation value patterns, 1s tend to add warp floats and 0s tend to add weft floats.

The "balanced" Boolean operations, which have two 1s and two 0s in their value patterns generally cause fewer problems with floats than the unbalanced ones. Note that there is only one balanced operation, *exclusive or*, in the functionally complete design set described earlier.

The problem of loom resources is more serious and difficult to gauge in design. One of Oelsner's examples of diversified weaves has 60 ends and 60 picks — and would require 60 shafts and 60 treadles (the maximum possible for a weave of this size). While this weave could be done on a drawloom, it really is in the province of Jacquard weaving.

Advanced Boolean Design

In this article, one Boolean operation has been applied to all the variables in two interlacement patterns.

The next article in this series describes the use of arrays of Boolean operations, in which different operations can be applied to different parts of the interlacement patterns.

References:

1. *An Investigation of the Laws of Thought on which are Founded the Mathematical Theories of Logic and Probabilities*, George Boole, 1854, Dover reprint, 1973.

2. *Boolean Algebra and its Applications*, J. Eldon Whitesitt, Addison-Wesley, 1961

3. *A Handbook of Weaves*, G. H. Oelsner, Macmillan, 1915. pp. 147.

Ralph E. Griswold

Department of Computer Science
The University of Arizona
Tucson, Arizona