

Designing Weave Structures Using Boolean Operations, Part 2

The first article on using Boolean operations in weave design [1] showed how a Boolean operation can be applied to arrays, source patterns, of Boolean variables that represent interlacement patterns to produce a new pattern.

The next step is to generalize this so that instead of applying just one Boolean operation to all variables, an array of Boolean operations is used, so that different operations apply to different parts of the two source patterns.

Arrays of Boolean Operations

Figure 1 shows an example of a 4×4 array with the Boolean operations *or* and *exclusive or*.

$$A = \begin{array}{cccc} \textit{xor} & \textit{or} & \textit{xor} & \textit{or} \\ \textit{or} & \textit{xor} & \textit{or} & \textit{xor} \\ \textit{xor} & \textit{or} & \textit{xor} & \textit{or} \\ \textit{or} & \textit{xor} & \textit{or} & \textit{xor} \end{array}$$

Figure 1. An Array of Boolean Operations

If we apply this array to a plain weave and a $2/2$ twill, we get the result shown in Figure 2.

$$\begin{array}{ccc} 1010 & 1100 & 0100 \\ 0101 & 0110 & 0011 \\ 1010 & 0011 & 1001 \\ 0101 & 1001 & 1100 \end{array} \quad A =$$

Figure 2. Applying the Array A

One of the problems with using arrays of Boolean operations is representing the arrays, especially when they are large.

Names, as in Figure 1, won't do: Although we can contrive names for all 16 Boolean operations on two variables, names take space and a large array of them would be huge and incomprehensible.

Using operator symbols instead of names saves space. The array of Figure 1 is shown in this way in Figure 3.

$$A = \begin{array}{cccc} \otimes & + & \otimes & + \\ + & \otimes & + & \otimes \\ \otimes & + & \otimes & + \\ + & \otimes & + & \otimes \end{array}$$

Figure 3. An Array of Boolean Operations

We picked an example for which there are operator symbols. We could invent operator symbols for the operations that don't already have them, but a large array with many different operator symbols would be incomprehensible also.

If we use the hexadecimal codes for the operations, we don't need extra symbols. See Figure 4.

$$A = \begin{array}{cccc} 6 & e & 6 & e \\ e & 6 & e & 6 \\ 6 & e & 6 & e \\ e & 6 & e & 6 \end{array}$$

Figure 4. An Array of Boolean Operations

This may or may not be an improvement. Mentally translating the hexadecimal codes into their logical equivalents is possible but difficult to do, especially in the context of creative design.






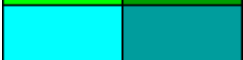


The human visual system is best at perceiving patterns when they are in color. We therefore have designed a color coding for the 16 Boolean operations.

These codes are largely arbitrary — there is no *a priori* mental relationship between colors and logical operations. Nevertheless, we can choose some colors that have natural interpretations, like white for the Boolean operation whose value pattern is 0000 and whose value is always the constant 0, which corresponds to a white cell in a drawdown. Simi-

larly, the Boolean operation with the value pattern 1111 and the constant value pattern 1 is naturally associated with black.

We've divided the remaining 14 operations into seven "basic" ones and their complements, choosing a different hue for each of the seven pairs. The "basic" operations have pure colors and their corresponding complements have corresponding dark shades.

Here are the colors we picked:

operation		color
0		white
x		blue
y		red
$x \times y$		magenta
$x + y$		green
$x \otimes y$		cyan
$x \rightarrow y$		brown
$x \leftarrow y$		gray

The designations x and y for operations indicate that the values of these operations depend only on their left and right operands, respectively.

Choosing bright colors for the basic operations sometimes produces garish arrays, but the important point is the ability to easily distinguish operations by colors, not to pro-

duce subtle, attractive patterns.

Figure 5 shows the color version of the array in Figures 1-4.

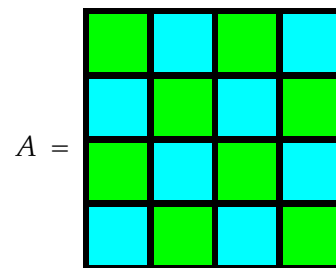


Figure 5. An Array of Boolean Operations

Figure 6 shows examples of the use of larger pattern and operation arrays.

Deficient Operations

As mentioned in the first article in this series, six of the 16 Boolean operations have values that do not depend on the values of both of their operands. We'll call these *deficient operations*. When only a single operation is applied to two patterns, deficient operations have little use.

When used in arrays of operations, however, the deficient operations can play important, even crucial roles.

The operations 0 and 1 force white and black cells, respectively, in the resulting patterns, providing a way to override the corresponding values in the source patterns. Figure 7 on the next page shows an example.

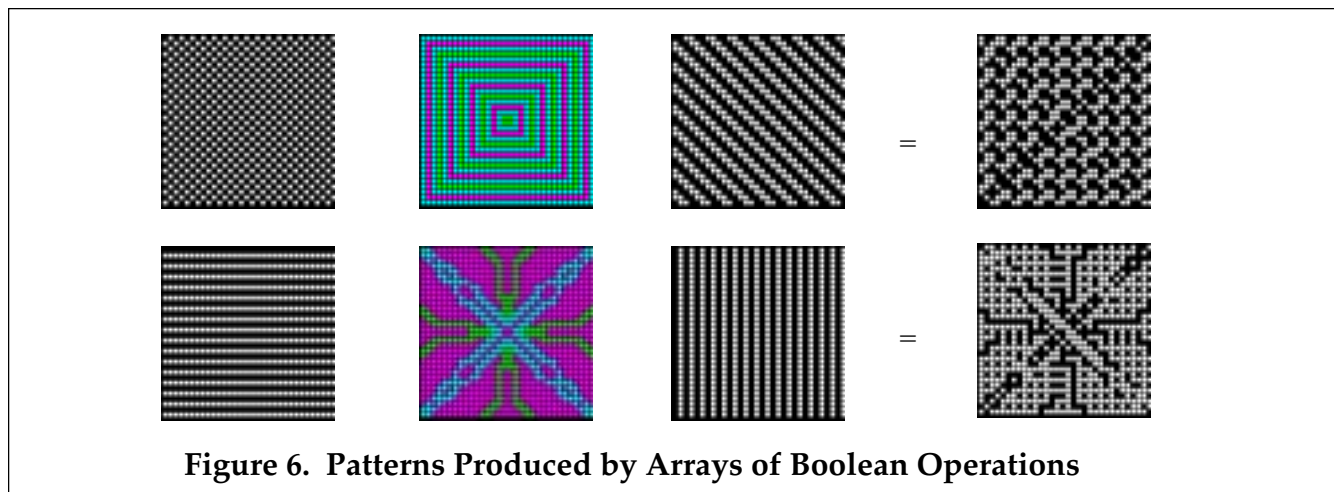


Figure 6. Patterns Produced by Arrays of Boolean Operations

The operations x and y force the left and right pattern values, respectively, to appear in the result. See Figure 8.

Comments

In articles on weave design, I try to present ideas and suggestions for unconventional design methods based on mathematical and computational tools.

For most of these, there are enormous ranges of possibilities. Boolean operations are perhaps the most promising of the methods we've described.

There is a problem, however. How do you actually apply the methods in practice? Applying arrays of Boolean operations to small interlacement patterns can be done by hand, although it is tedious and, perhaps more important, error prone. Large patterns and com-

plicated operation arrays are impractical to do by hand.

The general method for dealing with tasks that are infeasible to do manually is to use tools. Tools are, of course, the main reason for the achievements of the human race. The history of tools and tool making is long and complicated.

What tools might you design for applying arrays of Boolean operations? Perhaps you can come up with an approach using templates.

By far the most powerful tools for this kind of problem are computer programs.

In the next article on Boolean design, we'll describe two programs. One program allows Boolean operation arrays to be constructed interactively. The other program is for Boolean design itself.

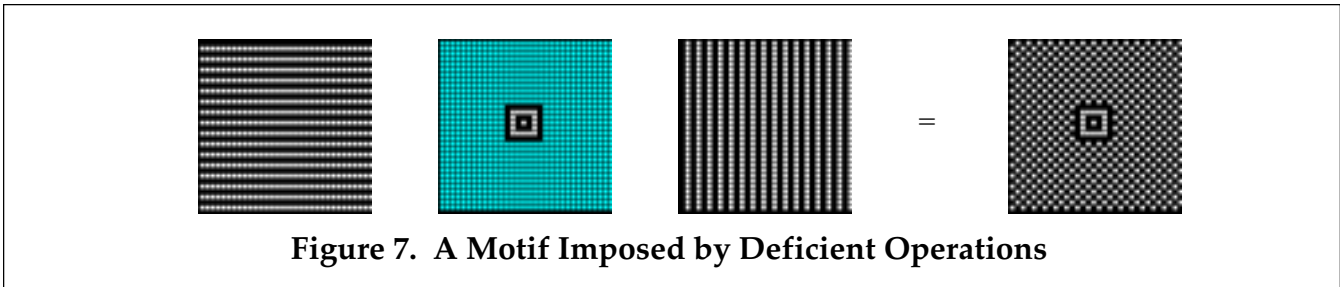


Figure 7. A Motif Imposed by Deficient Operations

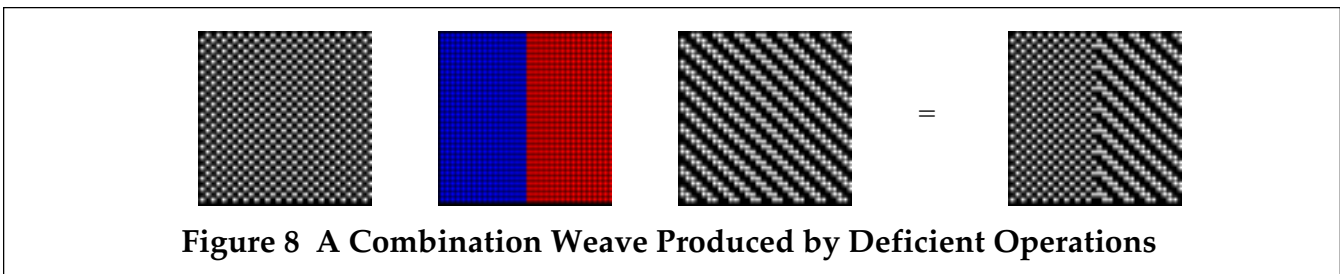


Figure 8 A Combination Weave Produced by Deficient Operations

Reference:

1. Ralph E. Griswold, "Designing Weave Structures Using Boolean Operations, Part 1", 2002: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_bol1.pdf

Ralph E. Griswold
 Department of Computer Science
 The University of Arizona
 Tucson, Arizona

© 2002, 2004 Ralph E. Griswold