Analysis of Weave Structures, Part 2: A Canonical Form

One issue that arises in fabric and pattern analysis is uniqueness. Is a pattern new or is it a simple transformation of another, previously recorded pattern.

What constitutes a transformation? Rotation? Reflection about an axis? Reversal of front and back? This is a matter of your point of view.

In the first article in this series [1], we assumed one transformation: repetition. That is, we reduced patterns to a fundamental motif when there were repeats. In other words, a pattern and repeats of a pattern are not considered to be essentially different.

Since most patterns are used in repeats, other transformations that generally are not taken to produce fundamentally different patterns are rotations (circular shifts) of rows and columns. When rotated patterns are repeated, the only noticeable effect is an allover shift in the design. It therefore makes sense to consider row and column rotations as "inessential" transformations.

This leaves the problem of how to determine if a pattern is the same as another except for row and column rotations. In some cases, it is visually obvious. In others, it is not.

A Canonical Form

This article describes a procedure for converting a pattern to a *canonical form*. This canonical form is the same for all row and column rotations of a pattern, enabling patterns equivalent in this sense to be determined. The procedure itself rotates rows and columns and in essence defines a fundamental pattern.

The basic idea behind this procedure is to find the row that is the "largest" in the sense of the first article on this subject.

In finding the largest row, all rotations of each row are performed, and the one that produces the largest result for that row is recorded. Then the largest of all these is the largest row for the entire pattern.

Whatever rotation produced the largest row is then applied to all rows (rotating columns).

This having been done, rows are rotated so that the largest is first.

The final step is to do the same things for the columns.

This description in words is imprecise and certainly hard to follow. A description in mathematical terms would require inventing notation and formalizing concepts for the case at hand and, while it would be more imposing than an informal description in words, it would be at least as incomprehensible.

Instead, we'll give an example in the hope that the ideas can be learned from it. For concreteness, there is an appendix with a computer procedure. This procedure is not meant to be comprehensible to persons that are not familiar with the programming language used [3], but it should give a general idea of the nature and difficulty of the process.

An Example

Figure 1 shows a pattern taken from Oelsner [2]:

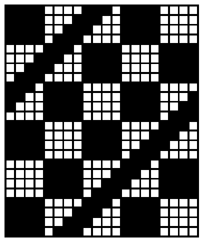

Figure 1. Pattern One

Figure 2 shows another pattern. Is it a rotated version of the first pattern?

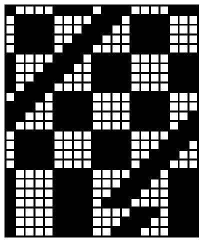

Figure 2. Pattern Two

You probably can determine that Pattern Two *is* a rotated version of Pattern 1. But can you be sure?

Let's start by getting the canonical form for Pattern 2.

The binary array for Pattern 2 is

```
10000111100111000111
10000111101110000111
01111000011101111000
01111000111001111000
01111001110001111000
01111011100001111000
10000111011110000111
10001110011110000111
10011100011110000111
10111000011110000111
01110111100001111000
11100111100001111000
11000111100001111001
10000111100001111011
01111000011110000111
01111000011110001110
01111000011110011100
01111000011110111000
10000111100001110111
10000111100011100111
10000111100111000111
10000111101110000111
10000111100001110111
10000111100011100111
```

Carrying out the procedure for getting the canonical form outlined above in all detail would start with the first line and form all 20 rotations of it to find the one that produces the (numerically) largest result (the most 1s at the left):

row	*amount*
10000111100111000111	0
00001111001110001111	1
00011110011100011110	2
…	
11000011110011100011	19

For this row, the largest result is obtained by a rotation of 5 and is 11110011100011110000. We do not need to do all these rotations to find this out, however. It is clear visually that the longest string of 1s that can result from any rotation is 4 with one followed by 0011 and the other followed by 0000. The first of these is the larger and a rotation by 5 brings it to the front, yielding the largest that can be had from this row.

The procedure would then do the same for all other rows. However, visual inspection shows that there is no row that can produce 5 1s in a row, regardless of the amount of rotation. Among the rows with 4 1s in a row, the largest has the string 11110111. This is found in first in row 2 with a rotation of 5 to bring this string into the lead position.

Therefore, all rows are rotated by 5 columns and all rows are rotated up by 1 to put the largest value in the first row.

For columns, we rotate the binary array by 90° and do the same thing. The final, canonical form for this pattern is shown in Figure 3.

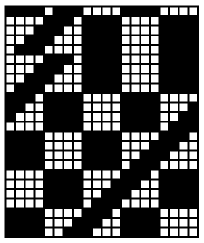

Figure 3. Canonical Form

The canonical form for Pattern One is the same. So Pattern Two is a rotated form of Pattern One and *vice versa*.

Since this pattern has 24 rows and 20 columns, there are 20 × 24 = 480 patterns that are equivalent under rotation and have this canonical form. Can you tell if any of these 480 patterns are *identical*? That is, are there rotations, other than 0, that don't change the pattern?

References

1. *Analysis of Weave Structures, Part 1: Introduction*, Ralph E. Griswold, 2004:
http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_ws1.pdf

2. *A Handbook of Weaves*, G. H. Oelsner, Dover reprint, page 117.

3. *The Icon Programming Language*, Ralph E. Griswold and Madge T. Griswold, 3rd ed., Peer-to-Peer, 1997:
http://www.cs.arizona.edu/icon/books.htm

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

© 2002, 2004 Ralph E. Griswold

Appendix

The following procedure, written in the Icon programming language, performs the row part of the canonicalization. The resulting array is the rotated 90° and the procedure is applied again to perform the column part of the process.

```
procedure canonic(rows)

   rows := copy(rows)

   max_rows := []
   max_rot := []

   every i := 1 to *rows do {
      row := rows[i]
      row_list := [row]
      every put(row_list, rotate(row, 1 to *rows – 1))
      put(max_rows, max_row := sort(row_list)[–1])
      put(max_rot, lindex(row_list, max_row))
      }

   max_row := sort(max_rows)[–1]

   j := lindex(max_rows, max_row) – 1
   p := max_rot[j + 1] – 1

   every i := 1 to *rows do
      rows[i] := rotate(rows[i], p)

   every 1 to j do
      put(rows, get(rows))

   return rows

end
```