From Drawdown to Draft — A Programmer's View

Nature uses only the longest threads to weave her patterns, so each small piece of her fabric reveals the organization of the entire tapestry.

— Richard Feynman

There are two distinct processes involved in analyzing the structure of a woven fabric:

- determining the interlacement of the warp and weft threads
- producing a draft from the interlacement

A particular method for doing fabric analysis may intermix these two processes, constructing the draft as the interlacement is determined. The two processes can be done separately, however, and there are advantages to separation:

- Determining the interlacement can be difficult. It requires a knowledge of weaving and careful visual examination of the fabric.
- Producing a draft from the interlacement is a mechanical task of an entirely different nature. It can be done by a person who is unfamiliar with weaving or by a computer program. A computer program is, of course, fast, but it offers a more important advantage: accuracy.

In this article, we'll describe a method of going from a drawdown to a draft and show how it can be done by a program.

Drawdowns

Various systems of notation are used for describing interlacement, but for the purposes of producing a draft, they are equivalent. The conventional drawdown grid, in which black squares indicate intersections where warp threads are on top and the white squares where weft threads are on top, is most widely used. Figure 1 shows an example.

Such a drawdown is designed to make it easy for a person to see the interlacement (and any patterns that may exist). For a computer program, a drawdown is just a rectangular array of zeros and ones with ones indicating where warp threads are on top and zeros indicating where weft threads are on top, as shown in Figure 2.

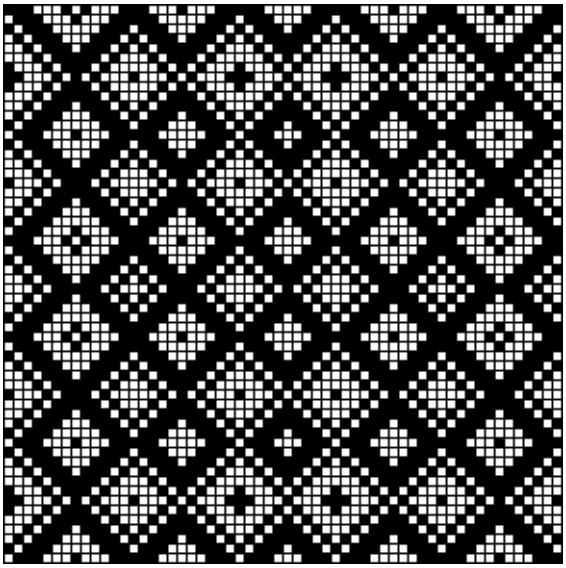

Figure 1. A Drawdown

```
11100001010001110001000111110000011111000100011100010100011
01110001000111110000011101110001110111000001111100001000111
10111000001110111000111010111011101011100011101110000011110
01011100001110101110111010101111101010111000111011100000111
00101110111010101111101000101110100010111111010101110111010
00010111111010001011101000001010100000101110100010111111010
10001011101000000101010001000101000100010101000001011101000
11000101010000010100011100010000111000101010001000101010001
10001011101000000101010001000101000100010101000001011101000
00010111111010001011101000001010100000101110100010111111010
00101110111010101111101000101110100010111111010101110111010
01011100011101011101110101011111010101110111010111011100011
10111000001110111000111010111011101011100011101110000011110
01110001000111110000011101110001110111000001111100001000111
10111000001110111000111010111011101011100011101110000011110
01011100011101011101110101011111010101110111010111011100011
10111000001110111000111010111011101011100011101110000011110
01110001000111110000011101110001110111000001111100001000111
11100001010001110001000111110000011111000100011100010100011
01110001000111110000011101110001110111000001111100001000111
10111000001110111000111010111011101011100011101110000011110
01011100011101011101110101011111010101110111010111011100011
00101110111010101111101000101110100010111111010101110111010
00010111111010001011101000001010100000101110100010111111010
10001011101000000101010001000101000100010101000001011101000
00010111111010001011101000001010100000101110100010111111010
00101110111010101111101000101110100010111111010101110111010
01011100011101011101110101011111010101110111010111011100011
10111000001110111000111010111011101011100011101110000011110
01110001000111110000011101110001110111000001111100001000111
11100001010001110001000111110000011111000100011100010100011
01110001000111110000011101110001110111000001111100001000111
10111000001110111000111010111011101011100011101110000011110
01011100011101011101110101011111010101110111010111011100011
00101110111010101111101000101110100010111111010101110111010
00010111111010001011101000001010100000101110100010111111010
10001011101000000101010001000101000100010101000001011101000
00010111111010001011101000001010100000101110100010111111010
00101110111010101111101000101110100010111111010101110111010
01011100011101011101110101011111010101110111010111011100011
10111000001110111000111010111011101011100011101110000011110
01110001000111110000011101110001110111000001111100001000111
11100001010001110001000111110000011111000100011100010100011
01110001000111110000011101110001110111000001111100001000111
10111000001110111000111010111011101011100011101110000011110
01011100011101011101110101011111010101110111010111011100011
00101110111010101111101000101110100010111111010101110111010
00010111111010001011101000001010100000101110100010111111010
10001011101000000101010001000101000100010101000001011101000
11000101010000010100011100010000111000101010001000101010001
10001011101000000101010001000101000100010101000001011101000
00010111111010001011101000001010100000101110100010111111010
00101110111010101111101000101110100010111111010101110111010
01011100011101011101110101011111010101110111010111011100011
10111000001110111000111010111011101011100011101110000011110
01110001000111110000011101110001110111000001111100001000111
```

Figure 2. Drawdown Data

Observations

The number of treadles required is the number of *different* row patterns. The number of shafts required is the number of *different* column patterns.

April 11, 2000; last revised September 15, 2004

In a large, complicated drawdown, it's difficult for a human being to determine the different row and column patterns. That's where a computer program comes in, as we'll show.

Comment: You can rearrange the rows and columns of a drawdown or delete duplicates without affecting the number of treadles and shafts required.

The Process

Creating a draft from a drawdown is a three-step process. The first two steps, which are central to the approach we are taking, can be done in either order.

The first step identifies the different rows and assigns a treadle number to each. The sequence of treadle numbers for the rows in the drawdown gives the treadling sequence.

The second step does the same thing for the columns to assign shaft numbers to the different columns and produce the threading sequence.

Comment: For the purposes of getting a workable draft, it doesn't matter which treadles and shafts are assigned to the different rows and columns. A systematic method, however, such as working from right to left or left to right, usually produces a better-organized draft.

The third step is to produce the tie-up that relates the treadles and shafts according to the drawdown.

To see how the first two steps might go by hand, look at figure 3, which shows treadle and shaft assignments to the upper-left portion of the drawdown we've been considering.

	shafts
treadles	**1 2 3 4 5 6 7 8 7 6 ...**
1	1 1 1 0 0 0 1 0 1 0 0 0 1 ...
2	0 1 1 1 0 0 0 1 0 0 0 1 1 ...
3	1 0 1 1 1 0 0 0 0 0 1 1 1 ...
4	0 1 0 1 1 1 0 0 0 1 1 1 0 ...
5	0 0 1 0 1 1 1 0 1 1 1 0 1 ...
6	0 0 0 1 0 1 1 1 1 1 1 0 1 0 ...
7	1 0 0 0 1 0 1 1 1 1 0 1 0 0 ...
8	1 1 0 0 0 1 0 1 0 1 0 0 0 ...
7	1 0 0 0 1 0 1 1 1 1 0 1 0 0 ...
6	0 0 0 1 0 1 1 1 1 1 1 0 1 0 ...
...	...

Figure 3. Treadle and Shaft Assignments

We can easily see that the beginnings of the first eight rows are different and therefore they are different patterns and each gets a separate treadle. The ninth row starts out like the seventh row and if we look at the complete drawdown in Figure 1 we can see they are the same. Similarly, the tenth row is the same as the sixth. The same is true of the columns.

We can continue in this fashion, but we have to be careful, because it's easy to get confused and make mistakes.

In this example, there are only eight different row patterns and eight different column patterns. They can be distinguished by the first three digits they contain. But another drawdown might be more complex, irregular, and not so easily analyzed by hand.

Programs

In this section, we'll show programs for determining the loom resources required and producing a draft from a drawdown.

You don't need to be a programmer to get an idea of what's going on. If you're not a programmer, just read through what follows and ignore the details.

How easy it is to write program to analyze a drawdown depends to a considerable extent on the programming language used. In our examples here, we'll use Icon [1, 2], a high-level programming language designed for the manipulation of strings of characters (like row and column patterns) and structures (like lists of patterns).

If you're a programmer but not familiar with Icon, just browse through what follows, note the comments, and imagine how you'd do it in your favorite programming language.

First, a word about strings and data structures in Icon. Strings are sequences of characters. A data structure is a collection of values that are organized in a particular way. We'll use three kinds of data structures in the programs that follow:

A *list* is a sequence of values. The values may be strings or other types of values.

A *set* is an unordered collection of unique values. A set can be created from a list; any duplicate values in the list are discarded.

A *table* is like a set, except that it has unique keys with which values can be associated.

Strings, lists, sets, and tables are created and modified as a program runs.

Computing Treadle and Shaft Requirements

Here's a little program that reads a draw-

down in the form of rows of zeros and ones such as shown in Figure 2. It writes out the number of treadles and shafts required.

```
procedure main()
   # Read the drawdown and put it in a list.
   rows := []                    # empty list to start
   while put(rows, read())       # add the row patterns
   # Write the number of treadles needed.
   write(*set(rows), " treadles needed" )
   # Rotate the drawdown 90 degrees to put the
   # columns in the place of rows.
   cols := rotate(rows)
   # Write the number of shafts needed.
   write(*set(cols), " shafts needed" )
end
```

The conversion of the list rows to a set, using set(rows), creates a set of rows without the duplicates. The operation * produces the number of members in the set. The procedure rotate() is shown in the complete program listing in Appendix A. The number of shafts required is then determined in the same way as the number of treadles.

The output of the program for the data shown in Figure 2 is

```
8 shafts needed
8 treadles needed
```

Comment: A drawdown must, of course contain at least one full repeat. If it contains more, the extra rows and columns are just duplicates and do not affect the result.

Producing a Draft

To produce a draft, a little more work is required. Here's a program.

```
procedure main()
   # Read the drawdown and put it in a list.
   rows := []                    # empty list to start
   while put(rows, read())       # add the row patterns
   cols := rotate(rows)          # list of columns
   # Compute the treadling sequence.
   number := 0                   # treadle counter
   treadles := table()           # table of row patterns
   # Build a table of the different row patterns and
   # assign a shaft number to each.
   every treadles[!set(rows)] := (number +:= 1)
   # Compute the threading sequence the same way.
   shafts := table()
   number := 0
   every shafts[!set(cols)] := (number +:= 1)
   # Create the tie–up.
   tieup := table()
   every row := key(treadles) do {
         tie_line := repl("0", *shafts) # no ties to start
      every i := 1 to *row do           # go through row
         if row[i] == "1" then          # tie if warp on top
            tie_line[threading[i]] := "1"  # for rising shed
      tieup[treadles[row]] := tie_line # add tie-up line
      }
   # Write the treadling sequence.
   write("Treadling sequence:")
   every writes(treadles[!rows], " ")
   write()
   # Write the threading sequence.
   write("Threadling sequence:")
   every writes(shafts[!cols], " ")
   write()
   # Write the tie–up.
      every i := 1 to *treadles do
      write(tieup[i])
end
```

The operator ! generates the members of a set. Each one becomes a key in a table, with which a number is associated (+:= 1 increments the value of number). To produce the sequences, the rows and columns are generated and used as keys to the tables, which in turn produces the numbers.

The tie-up is created using a table. key(treadles) produces the row patterns. Each line of the tie-up starts with zeros, indicating the absence of ties. Then for every position of the row pattern that is one, the position in the row of the corresponding shaft is set to one to indicate a tie.

The sequences are produced from left to right, since that is the most natural and easiest way to program the process.

The output for the data shown in Figure 2 is:

```
Treadling sequence:
1 2 3 4 5 6 7 8 7 6 5 4 3 2 3 4 5 6 7 6 5 4 3 2 1 2 3 4
5 6 5 4 3 2 1 2 3 4 5 6 7 6 5 4 3 2 3 4 5 6 7 8 7 6 5 4
3 2
Threading sequence:
1 2 3 4 5 6 7 8 7 6 5 4 3 2 3 4 5 6 7 6 5 4 3 2 1 2 3 4
5 6 5 4 3 2 1 2 3 4 5 6 7 6 5 4 3 2 3 4 5 6 7 8 7 6 5 4
3 2
Tie-up:
11100010
01110001
10111000
01011100
00101110
00010111
10001011
11000101
```

Note that the treadling and threading sequences are the same: The draft is treadled as drawn.

Figure 4 shows a conventional draft produced from the results given by the program above.

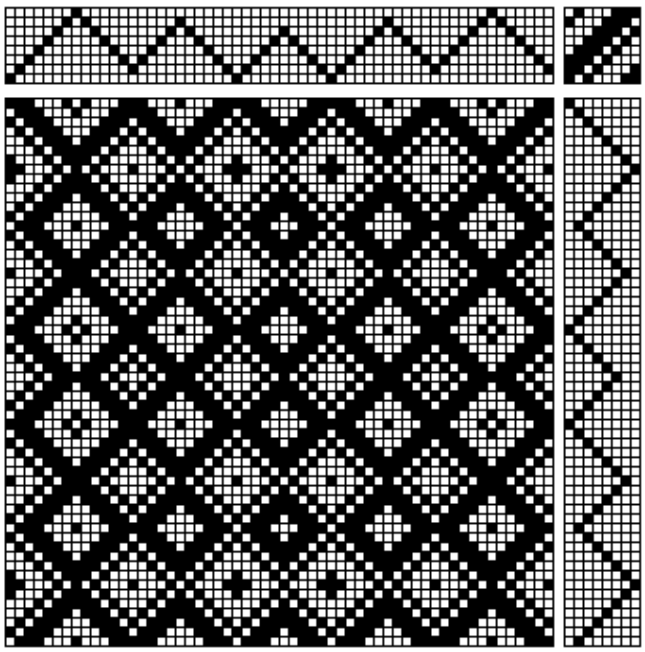

Figure 4. The Draft

Related Issues

WIFs

The program that produces the draft information can be modified to produce a WIF [3], which can be imported into a weaving program and exchanged with other weavers.

This is not difficult to do, but WIF is a verbose format and the code needed to produce a WIF is considerably longer than the code to analyze the drawdown.

The Appendix A shows the complete program for going from the raw data to a WIF. Appendix B shows the WIF for the data shown in Figure 2.

Drafts from Images

Any black and white image* can be considered to be a drawdown — the black pixels (individual dots) correspond to where imaginary warp threads are on top and the white pixels correspond to where imaginary weft threads are on top.

To get a draft for weaving the equivalent of such an image, it only is necessary to convert the pixels in the image to patterns of zeroes and ones as shown in Figure 2.

This requires a little bit of computer graphics. In Icon, it looks like this [4]:

```
WOpen("image=design.gif") # window for image

width := WAttrib("width")       # dimensions
height := WAttrib("height")

rows := []

# Get the row patterns

every y := 0 to height – 1 do {
  row := ""
  every p := Pixel(0, y, width, 1) do
    if ColorValue(p) == "0,0,0" then row ||:= "1"
      else row ||:= "0"
  put(rows, row)
  }
```

The value "0,0,0" corresponds to a black pixel. Other pixels are assumed to be white. The rest of the program is the same as before.

Comment: Some weaving programs, such as SwiftWeave [5], provide a facility for going from black-and-white images to drafts. SwiftWeave calls it a drawup.

A draft created from an image can be used as a profile draft or a threading draft. For a threading draft, modifications may be needed to make it weavable.

Although any two-color pattern can be converted to a draft, the problem is the loom resources

*In fact, any two-color image can be considered to be a drawdown; simply change one color to black for the warp and the other to white for the weft.

required — the number of shafts and treadles — which exceed the capacity of your loom — or any loom (recall that the number of treadles required is the number of different rows and the number of shafts required is the number of different columns).

If you want to try this, pick images that are small, with straight lines, and lots of duplicate rows and columns. Horizontal and vertical symmetry come free.

Appendix C shows an image and the drawdown produced by this method.

On-Line Resources

The programs described here, along with sample data and images, are available on the Web [6].

References

1. *The Icon Programming Language*, third edition, Ralph E. Griswold and Madge T. Griswold, Peer-to-Peer Communications, Inc., 1996:
http://www.cs.arizona.edu/icon/books.htm

2. *The Icon Programming Language:*
http://www.cs.arizona.edu/icon/

3. *WIF Specification:*
http://www.mhsoft.com/wif/wif.html

4. *Graphics Programming in Icon*, Ralph E. Griswold, Clinton L. Jeffery, and Gregg M. Townsend , Peer-to-Peer Communications, Inc., 1998.

5. *Swiftweave:*
http://www.swiftweave.com/

6. *Weaving Programs:*
http://www.cs.arizona.edu/patterns/weaving/programs.html

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

Appendix A — WIFs from Drawdowns

```
procedure main()

   # Read the drawdown and put it in a list.

   rows := []                    # empty list to start

   while put(rows, read())       # add the row patterns

   cols := rotate(rows)          # list of columns

   # Compute the treadling sequence.

   number := 0                   # treadle counter
   treadles := table()           # table of row patterns

   # Build a table of the different row patterns and
   # assign a shaft number to each.

   every treadles[!set(rows)] := (number +:= 1)

   # Compute the threading sequence the same way.

   shafts := table()
   number := 0

   every shafts[!set(cols)] := (number +:= 1)

   # Create the tie–up.

   tieup := table()
  every row := key(treadles) do {
         tie_line := repl("0", *shafts)      # no ties to
start
      every i := 1 to *row do          # go through row
        if row[i] == "1" then          # tie if warp on top
          tie_line[threading[i]] := "1"
      tieup[treadles[row]] := tie_line # add tie–up line
```


Appendix A — WIFs from Drawdowns

```
procedure main()

   # Read the drawdown and put it in a list.

   rows := []                    # empty list to start

   while put(rows, read())       # add the row patterns

   cols := rotate(rows)          # list of columns

   # Compute the treadling sequence.

   number := 0                   # treadle counter
   treadles := table()           # table of row patterns

   # Build a table of the different row patterns and
   # assign a shaft number to each.

   every treadles[!set(rows)] := (number +:= 1)

   # Compute the threading sequence the same way.

   shafts := table()
   number := 0

   every shafts[!set(cols)] := (number +:= 1)

   # Create the tie–up.

   tieup := table()
  every row := key(treadles) do {
         tie_line := repl("0", *shafts)      # no ties to
start
      every i := 1 to *row do          # go through row
        if row[i] == "1" then          # tie if warp on top
          tie_line[threading[i]] := "1"
      tieup[treadles[row]] := tie_line # add tie–up line
```


```
   }

   #  Now output the WIF.

   write("[WIF]")
   write("Version=1.1")
   write("Date=" || &dateline)
   write("Developers=ralph@cs.arizona.edu")
   write("Source Program=dd2wif.icn")
   write("[CONTENTS]")
   write("Color Palette=yes")
   write("Text=yes")
   write("Weaving=yes")
   write("Tieup=yes")
   write("Color Table=yes")
   write("Threading=yes")
   write("Treadling=yes")
   write("Warp colors=yes")
   write("Weft colors=yes")
   write("Warp=yes")
   write("Weft=yes")
   write("[COLOR PALETTE]")
   write("Entries=", 2)
   write("Form=RGB")
   write("Range=0,65535)
   write("[TEXT]")
   write("Title=example")
   write("Author=Ralph E. Griswold")
  write("Address=5302 E. 4th St., Tucson, AZ 85711")
   write("EMail=ralph@cs.arizona.edu")
   write("Telephone=520-881–1470")
   write("FAX=520–325–3948")
   write("[WEAVING]")
   write("Shafts=", *shafts)
   write("Treadles=", *treadles)
   write("Rising shed=yes")
   write("[WARP]")
```

```
   write("Threads=", *threading)
   write("Units=Decipoints")
   write("Thickness=10")
   write("Color=1")
   write("[WEFT]")
   write("Threads=", *treadling)
   write("Units=Decipoints")
   write("Thickness=10")
   write("Color=2")
   write("[WARP THICKNESS]")
   write("[WEFT THICKNESS]")
   write("[COLOR TABLE]")
   write("1=0,0,0")
   write("2=65535,65535,65535")
   write("[THREADING]")
   every i := 1 to *threading do
      write(i, "=", threading[i])
   write("[TREADLING]")
   every i := 1 to *treadling do
      write(i, "=", treadling[i])
   write("[TIEUP]")
   every i := 1 to *tieup do
      write(i, "=", tromp(tieup[i]))
end

procedure tromp(treadle)
   result := ""
   every i := 1 to *treadle do
      if treadle[i] == "1" then result ||:= i || ","
   return result[1:–1]    # remove trailing comma
end

procedure rotate(rows)
   cols := list(*rows[1], "")
   every row := !rows do {
      i := 0
      every grid := !row do
         cols[i +:= 1] := grid || cols[i]
      }
   return cols
end
```

Appendix B — WIF Output

```
[WIF]
Version=1.1
Date=Tuesday, April 11, 2000  11:25 am
Developers=ralph@cs.arizona.edu
Source Program=dd2wif.icn
[CONTENTS]
Color Palette=yes
Text=yes
Weaving=yes
Tieup=yes
Color Table=yes
Threading=yes
Treadling=yes
Warp colors=yes
Weft colors=yes
Warp=yes
Weft=yes
[COLOR PALETTE]
Entries=2
Form=RGB
Range=0,65535
[TEXT]
Title=example
Author=Ralph E. Griswold
Address=5302 E. 4th St., Tucson, AZ 85711
EMail=ralph@cs.arizona.edu
Telephone=520–881–1470
FAX=520–325–3948
[WEAVING]
Shafts=8
Treadles=8
Rising shed=yes
[WARP]
Threads=58
Units=Decipoints
Thickness=10
Color=1
[WEFT]
Threads=58
Units=Decipoints
Thickness=10
Color=2
[WARP THICKNESS]
[WEFT THICKNESS]
[COLOR TABLE]
1=0,0,0
2=65535,65535,65535
[THREADING]
1=1
2=2
3=3
```

```
4=4
5=5
        …       (lines omitted)
54=6
55=5
56=4
57=3
58=2
[TREADLING]
1=1
2=2
3=3
4=4
5=5
        …       (lines omitted)
54=6
55=5
56=4
57=3
58=2
[TIEUP]
1=1,2,3,7
2=2,3,4,8
3=1,3,4,5
4=2,4,5,6
5=3,5,6,7
6=4,6,7,8
7=1,5,7,8
8=1,2,6,8
```

Appendix C – Draft from Image

Figure 1 shows a Japanese repeat pattern. It is 65 by 65 pixels.

Figure 1 A Japanese Pattern

The resulting draft requires 32 shafts and 32 treadles. See Figure 2.

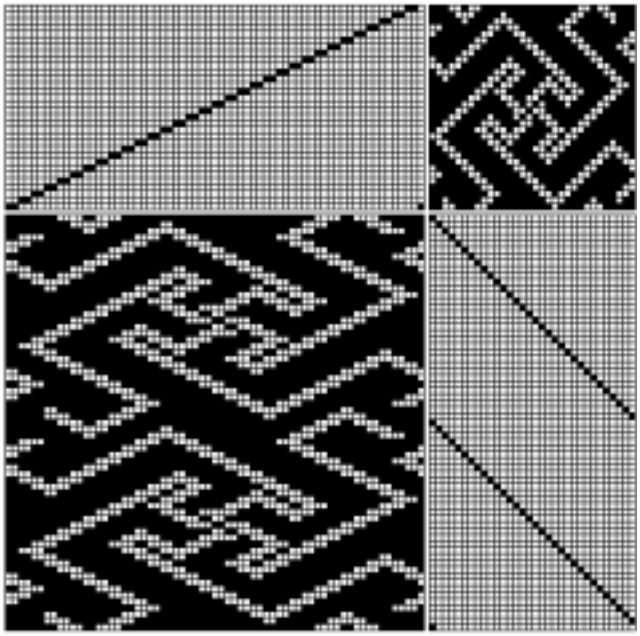

Figure 2. Draft for Japanese Pattern