



Constrained Patterns

Constraints limit what is possible. With respect to interlacement patterns, constraints impose both color and structural limitations. Constraints can take many forms. Expressed in terms of drawdowns, examples are:

1. The number of white cells and black cells must be equal.
2. No more than four consecutive cells in any row and column can be the same color.
3. Every cell must have at least one adjacent cell of the opposite color.
4. Constraints 1, 2, and 3 all must be satisfied.

Constraint 1 is a global constraint and is equivalent to requiring that a weave be balanced. This constraint cannot be satisfied by a drawdown with an odd number of cells. That is, of all drawdowns, only ones with even dimensions can possibly satisfy this constraint.

Constraint 2 is more local and in more familiar terms limits float length.

Constraint 3 is local; it specifies a property that all neighborhoods must have. [\[Cross-reference cellular automata.\]](#)

Constraint 4 requires that three constraints be simultaneously satisfied. It is called a *constraint set*. In this sense, Constraints 1, 2, and 3 are constraint sets containing only one constraint.

Constraint Analysis

Given a pattern, it generally is easy to determine if it satisfies a given constraint set. For example, whether or not a pattern satisfies Constraint Set 1 can be determined just by counting black and white cells. Similarly, whether or not a pattern satisfies Constraint Set 2 can be determined by examination or using the float-analysis feature of a weaving program.

Constraint Set 3 requires a little more work, since it may be necessary, in general, to examine a large number of individual neighborhoods.

And, of course, determining whether or not a pattern satisfies Constraint Set 4 can be determined by checking each of the constraints in its constraint set.

It is important to realize that there are constraint sets that no patterns satisfy. For example, a constraint set that contains Constraint 1 and a constraint that patterns must have an odd number of cells cannot be satisfied — it is *unsatisfiable*. In this example, it is obvious that the two constraints are mutually exclusive. In general, it may be difficult to determine whether or not a constraint set can be satisfied any pattern.

Neighborhood Constraints

As far as weave structure is concerned, neighborhood constraints are interesting, since they have a strong effect on appearance.

Neighborhood constraints can be characterized by *neighborhood templates*. As with drawdown automata, there are many kinds of neighborhoods that can be used. The von Neumann 5-cell neighborhood [2] is used in what follows. This neighborhood is small enough to be computationally tractable but large enough to characterize a wide range of structural characteristics.

Neighborhood constraints can be pictured like this:



None of these constraints *taken alone* is satisfiable, simply because they all require every cell to be the same color (white in the first, black in the other two) while simultaneously requiring that them to be surrounded by cells of other colors.

Taken in combination in constraint sets, however, they may be satisfiable. For example, the constraint set consisting of the templates



is satisfied by plain weave (and only plain weave).

On the other hand, the constraint set consisting of the templates



is unsatisfiable because it requires every cell to be black and at the time to have adjacent white cells.

Neighborhood constraints can be looked at in several ways:

- Does a pattern satisfy a given neighborhood constraint set?
- What neighborhood constraint set does a given pattern satisfy?
- What patterns satisfy a specific neighborhood constraint set?

The first question is easy to answer: as mentioned above, it's only necessary to compare the cell neighborhoods to the templates in the constraint set.

The second question also is generally easy to answer by cataloging the



neighborhoods of all cells, although there are some issues to be addressed, such as how to handle cells at the borders that do not have complete neighborhoods.

The third question is, in general, much harder to answer. It is, nonetheless, interesting. For example, it would be interesting to know what patterns satisfy the same neighborhood constraint set that a 2/2 twill does. The problem is hard because there is no known way to construct patterns from constraint sets that does not involve a large amount of computation.

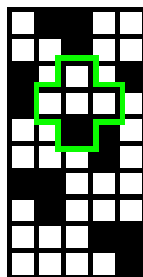
Neighborhood Analysis

In the first article on constraints, we introduced the concept of neighborhood constraints [1]. In this article, we'll look at the problem of determining the neighborhood constraint set of a pattern.

Consider the following pattern:



All that is necessary to determine the constraint set for this pattern is to examine every cell and record the template for its neighborhood.



For example, the template for the cell outlined above is

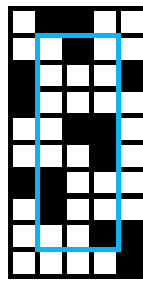


This process is straightforward except for cells at the edges, which have incomplete neighborhoods. There are several ways to handle such cells:



1. Don't include the edge cells in the analysis.
2. Assume that the pattern repeats so that the edges wrap around.
3. Don't assume the pattern repeats (for example, the Morse-Thue carpet does not [2]) but include partial neighborhoods of the edge cells.

Method 1 amounts to analyzing a sub-pattern, shown by the blue outline below:

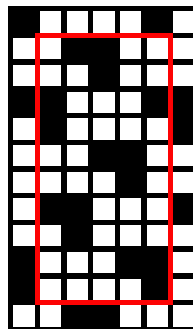


The problem with this approach is that the constraint set obtained may not be complete. For example, the unit motif for plain weave is a 2×2 pattern:



This pattern only has edge cells. If they are ignored, there is no constraint set at all, which obviously is incorrect.

Method 2 can be handled by augmenting the pattern, adding cells around the edges that correspond to what would appear if the pattern were contained in a repeat:



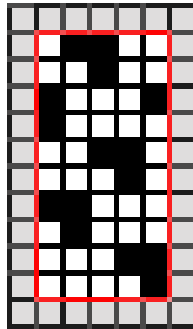
Now the analysis can proceed for the cells enclosed in the red rectangle above;



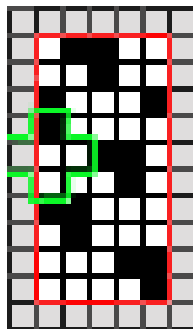
effectively there are no edge cells.

This method is fine for repeating patterns, but it produces erroneous results for aperiodic patterns such as the Morse-Thue carpet.

Method 3 tries to deal with this situation by adding edges with unknown cell colors:



In this case, an edge cells such as the one outlined below has a neighborhood template with a cell of unspecified color:

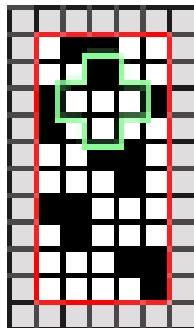


Here is that template:



It can be added this partial constraint to the set. But note that there is other cells in the pattern with complete templates that have the same three cells as the partial constraint:





This neighborhood,



“covers” the incomplete one, it is not necessary to keep the incomplete one.

If partial constraints remain after analyzing all cells, one possibility is to just “force them” by arbitrarily coloring the unspecified cells.

What to do about an aperiodic patterns is an open question. One can analyze a portion of it using Method 3. But how can one tell if the constraint set obtained is complete? Would analyzing a larger portion add to the constraint set?

In the case of the Morse-Thue carpet, analyzing a modest portion yields a constraint set with 18 templates. Analyzing larger portions do not increase the size of the constraint set. It seems reasonable, examining the method by which the Morse-Thue carpet is constructed, that this constraint set applies to the entire, unlimited pattern.

But for other patterns, such as random ones, there is no basis for such an assumption. In fact, the constraint set for a randomly generated pattern may include all 32 possible constraints.

On the other hand, what is the point of trying to determine the neighborhood constraint set for a pattern without structure?

Representing Constraint Sets

Neighborhood constraint sets can be represented in several ways. For human understanding, graphical methods work best. For computer processing, textual representations or numerical codes are more appropriate.



Graphical Representations

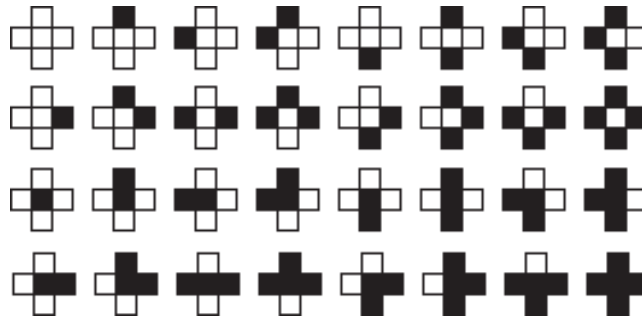
In previous articles, we showed templates as neighborhoods laid out according to their natural geometrical interpretation, as in



Any constraint set then can be represented by a collection of such template images. For example, the constraint set for plain weave is



If a constraint set is large, this kind of representation takes a fair amount of space for images of a size sufficient to be readily understood. For example, the constraint set containing all constraints is



A less useful but more compact graphical representation is as a bar of 32 cells, each cell corresponding to one of the constraints. If a cell is in a constraint set, it is colored gray, otherwise white. Gray is used so the black dividing lines can be used as a guide to cell position. For example, the cell bar for the plain-weave constraint set is



The problem with the cell-bar representation is that the templates are coded by position, so that to determine the constraints, it's necessary to know where individual templates are in the bar and the order of the templates (which is as shown in the image for all templates, reading left to right and top to bottom). Determining the actual templates in this way is tedious and error prone, so the cell-bar representation is not appropriate for that purpose. It is suitable, however, for getting an idea of the number of constraints in a set and comparing

patterns of different constraint sets.

Textual Representations

The graphic representation as a series of templates has a natural counterpart as a list of 5-bit binary strings in which a bit is 1 if the corresponding cell in the neighborhood is black and 0 otherwise.

A convention is needed to determine the order of the bits in the bit string. The convention we'll use here numbers the cells starting with the center cell and continuing clockwise around the outer cells:

```

      5
    4 1 2
      3
  
```

Therefore the plain-weave constraint set has the textual representation

```

01111
10000
  
```

(Since this represents a set, the order of the constraints is not important, but a useful convention is to order the binary strings by magnitude, as we have done in this example.)

A more compact textual representation of constraint sets is as 32-bit binary strings in which a bit is one if the corresponding constraint is in the set and 0 otherwise. For example, the plain-weave constraint set represented in this way is

```
00000000000000000000000000000000
```

Note that although the cell-bar representation is difficult for a human being to interpret in its entirety, the 32-bit binary string representation presents no problem for a program: It's just another decoding task of the kind that programs have to handle all the time.

Numerical Codes

A variation on the textual representations is to think of bit strings as base-2 integers. These base-2 integers then can be converted to conventional base-10 integers. For example, in terms of 5-bit constraints, the plain-weave constraint set has the numerical codes

```

15
16
  
```




while the 32-bit representation has the numerical code 98305.

For computer programs, these are just other ways of encoding and present no more problems than the textual forms. Base-10 integers have advantages for programming in some situations because all commonly used programming languages support integer arithmetic. However, the equivalents of 32-bit bit strings can be very large: as large as 4,294,967,296, which is beyond the range of integer arithmetic in most programming languages.

Numerical codes are somewhere between graphical representations, suitable for human beings, and textual representations, suitable for computers. For human beings, they do have value as labels, if arbitrary, and are only about one-third the length of the corresponding binary strings, as well as being easier to differentiate than bit strings.

von Neumann Constraint Pattern Catalog

[This section has not been published on the Web.]

Wolfram [1] has shown that only 171 repeating patterns are needed to characterize all the von Neumann neighborhood constraint sets [2]. (Rotations, reflections, and color reversals are omitted.)




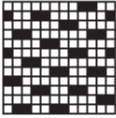












The pages that follow show unit weaves for these patterns in the order given in Reference 1. Below each pattern are its dimensions and, if weavable from a drawup, the loom resources required. At the bottom is a cell bar showing the constraints involved [3].

Some of the patterns that are unweavable as drawups can be drafted using color-and-weave effects. Obvious examples are the stripes.

















All of the patterns that are weavable from drawn-up drafts “hang together” [4]. [Cross reference; consider changing “weavable” to “draftable” here and elsewhere.]



190

<div>1</div> <div></div> <div>1x1</div>	<div>2</div> <div></div> <div>5x5 5x5</div>	<div>3</div> <div></div> <div>2x4</div>	<div>4</div> <div></div> <div>12x12 12x12</div>
<div>5</div> <div></div> <div>1x3</div>	<div>6</div> <div></div> <div>6x2</div>	<div>7</div> <div></div> <div>1x2</div>	<div>8</div> <div></div> <div>3x4</div>
<div>9</div> <div></div> <div>3x3 3x3</div>	<div>10</div> <div></div> <div>6x3 3x3</div>	<div>11</div> <div></div> <div>5x5 5x5</div>	<div>12</div> <div></div> <div>4x4 4x4</div>
<div>13</div> <div></div> <div>5x10 5x10</div>	<div>14</div> <div></div> <div>3x9</div>	<div>15</div> <div></div> <div>8x8 8x8</div>	<div>16</div> <div></div> <div>4x4 4x4</div>



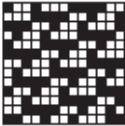







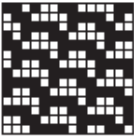


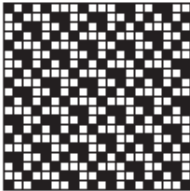
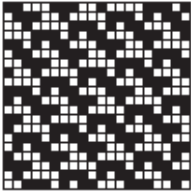



17  3x3	18  9x9 9x9	19  3x12	20  4x4 4x4
21  8x4 4x4	22  5x4 5x4	23  4x8 4x8	24  7x7 7x7
25  5x10 5x10	26  4x3 3x3	27  9x3 6x3	28  12x3 6x3
29  6x6 6x6	30  3x2	31  4x2 2x2	32  10x10 10x10



192



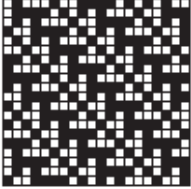
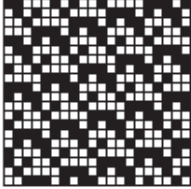

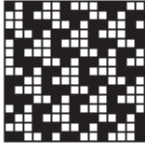







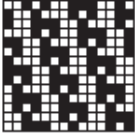


<div>33</div> <div></div> <div>3x6 3x6</div>	<div>34</div> <div></div> <div>7x14 7x14</div>	<div>35</div> <div></div> <div>4x3</div>	<div>36</div> <div></div> <div>3x3</div>
<div>37</div> <div></div> <div>8x8 8x8</div>	<div>38</div> <div></div> <div>9x18 9x18</div>	<div>39</div> <div></div> <div>3x5</div>	<div>40</div> <div></div> <div>11x11 11x11</div>
<div>41</div> <div></div> <div>15x5 15x5</div>	<div>42</div> <div></div> <div>15x5 15x5</div>	<div>43</div> <div></div> <div>6x9 6x9</div>	<div>44</div> <div></div> <div>12x4 10x4</div>
<div>45</div> <div></div> <div>17x17 17x17</div>	<div>46</div> <div></div> <div>3x9 3x9</div>	<div>47</div> <div></div> <div>16x20 16x20</div>	<div>48</div> <div></div> <div>12x3</div>






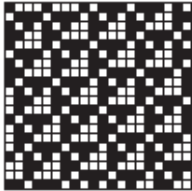












<p>49</p>  <p>13x13 13x13</p>	<p>50</p>  <p>15x5 10x5</p>	<p>51</p>  <p>10x5 10x5</p>	<p>52</p>  <p>4x16 4x16</p>
<p>53</p>  <p>10x10 10x10</p>	<p>54</p>  <p>5x15 5x15</p>	<p>55</p>  <p>7x20 7x20</p>	<p>56</p>  <p>20x7 20x7</p>
<p>57</p>  <p>14x14 14x14</p>	<p>58</p>  <p>4x6 4x6</p>	<p>59</p>  <p>4x3 3x3</p>	<p>60</p>  <p>20x20 20x20</p>
<p>61</p>  <p>20x20 20x20</p>	<p>62</p>  <p>5x10 5x10</p>	<p>63</p>  <p>15x3</p>	<p>64</p>  <p>14x14 14x14</p>



194



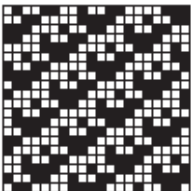






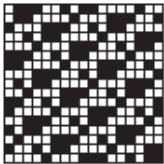






<p>65</p>  <p>7x14 7x14</p>	<p>66</p>  <p>12x4 12x4</p>	<p>67</p>  <p>20x20 16x20</p>	<p>68</p>  <p>20x20 20x20</p>
<p>69</p>  <p>14x7 14x7</p>	<p>70</p>  <p>15x15 15x15</p>	<p>71</p>  <p>7x14 7x14</p>	<p>72</p>  <p>9x18 9x18</p>
<p>73</p>  <p>6x6 6x6</p>	<p>74</p>  <p>20x9 18x9</p>	<p>75</p>  <p>2x2 2x2</p>	<p>76</p>  <p>3x3 2x3</p>
<p>77</p>  <p>16x20 16x20</p>	<p>78</p>  <p>14x14 14x14</p>	<p>79</p>  <p>15x3 6x3</p>	<p>80</p>  <p>12x3 6x3</p>














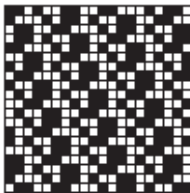




<p>81</p>  <p>17x20 17x20</p>	<p>82</p>  <p>5x15 5x15</p>	<p>83</p>  <p>16x8 16x8</p>	<p>84</p>  <p>20x20 20x20</p>
<p>85</p>  <p>14x14 14x14</p>	<p>86</p>  <p>4x6</p>	<p>87</p>  <p>6x3</p>	<p>88</p>  <p>7x14 7x14</p>
<p>89</p>  <p>12x3</p>	<p>90</p>  <p>11x11 11x11</p>	<p>91</p>  <p>10x4</p>	<p>92</p>  <p>5x15 5x15</p>
<p>93</p>  <p>8x8 8x8</p>	<p>94</p>  <p>18x9 18x9</p>	<p>95</p>  <p>9x20 9x20</p>	<p>96</p>  <p>20x20 20x20</p>



196






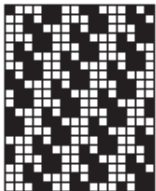



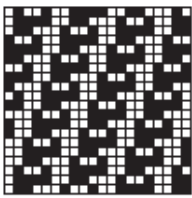






<p>97</p>  <p>4x5</p>	<p>98</p>  <p>7x14 7x14</p>	<p>99</p>  <p>20x20 20x20</p>	<p>100</p>  <p>6x9 6x9</p>
<p>101</p>  <p>12x12 12x12</p>	<p>102</p>  <p>7x14 7x14</p>	<p>103</p>  <p>15x5 10x5</p>	<p>104</p>  <p>13x13 13x13</p>
<p>105</p>  <p>20x11 20x11</p>	<p>106</p>  <p>17x17 17x17</p>	<p>107</p>  <p>6x6 6x6</p>	<p>108</p>  <p>6x4 6x4</p>
<p>109</p>  <p>9x18 9x18</p>	<p>110</p>  <p>20x9 18x9</p>	<p>111</p>  <p>10x10 10x10</p>	<p>112</p>  <p>12x12 12x12</p>




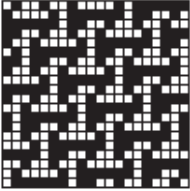



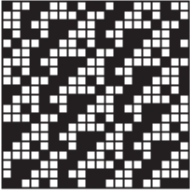




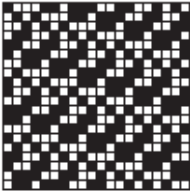





<p>113</p>  <p>7x14 7x14</p>	<p>114</p>  <p>4x8 4x8</p>	<p>115</p>  <p>20x20 20x20</p>	<p>116</p>  <p>15x3</p>
<p>117</p>  <p>8x16 8x16</p>	<p>118</p>  <p>18x3</p>	<p>119</p>  <p>15x3</p>	<p>120</p>  <p>4x8 4x8</p>
<p>121</p>  <p>5x15 5x15</p>	<p>122</p>  <p>4x20 4x20</p>	<p>123</p>  <p>4x12</p>	<p>124</p>  <p>20x20 17x20</p>
<p>125</p>  <p>14x7 14x7</p>	<p>126</p>  <p>4x12 4x12</p>	<p>127</p>  <p>3x6</p>	<p>128</p>  <p>14x7 14x7</p>













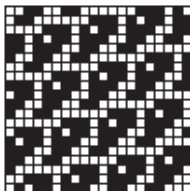



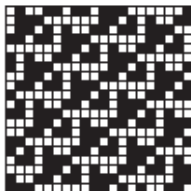







198

<p>129</p>  <p>13x13 13x13</p>	<p>130</p>  <p>20x20 18x20</p>	<p>131</p>  <p>6x4 6x4</p>	<p>132</p>  <p>20x8 20x8</p>
<p>133</p>  <p>20x4 14x4</p>	<p>134</p>  <p>16x20 16x20</p>	<p>135</p>  <p>20x17 17x17</p>	<p>136</p>  <p>13x13 13x13</p>
<p>137</p>  <p>9x20 9x20</p>	<p>138</p>  <p>20x20 17x20</p>	<p>139</p>  <p>16x4 10x4</p>	<p>140</p>  <p>8x8 4x8</p>
<p>141</p>  <p>12x4 5x4</p>	<p>142</p>  <p>12x6 12x6</p>	<p>143</p>  <p>7x14 7x14</p>	<p>144</p>  <p>17x20 17x20</p>



<p>145</p>  <p>20x20 18x20</p>	<p>146</p>  <p>20x20 20x20</p>	<p>147</p>  <p>20x11 20x11</p>	<p>148</p>  <p>20x5 13x5</p>
<p>149</p>  <p>20x17 17x17</p>	<p>150</p>  <p>20x20 20x20</p>	<p>151</p>  <p>20x11 20x11</p>	<p>152</p>  <p>7x14 7x14</p>
<p>153</p>  <p>4x10 4x10</p>	<p>154</p>  <p>18x6 15x6</p>	<p>155</p>  <p>20x20 20x20</p>	<p>156</p>  <p>4x10 4x10</p>
<p>157</p>  <p>5x20 5x20</p>	<p>158</p>  <p>10x6 4x6</p>	<p>159</p>  <p>16x16 16x16</p>	<p>160</p>  <p>18x9 18x9</p>



<div>161</div> <div></div> <div>11x20 11x20</div> <div></div>	<div>162</div> <div></div> <div>12x6 9x6</div> <div></div>	<div>163</div> <div></div> <div>7x14 7x14</div> <div></div>	<div>164</div> <div></div> <div>20x10 20x10</div> <div></div>
<div>165</div> <div></div> <div>4x16 4x16</div> <div></div>	<div>166</div> <div></div> <div>20x20 20x20</div> <div></div>	<div>167</div> <div></div> <div>6x9 6x9</div> <div></div>	<div>168</div> <div></div> <div>20x20 20x20</div> <div></div>
<div>169</div> <div></div> <div>20x11 20x11</div> <div></div>	<div>170</div> <div></div> <div>4x1</div> <div></div>	<div>171</div> <div></div> <div>3x18</div> <div></div>	