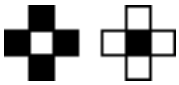
Constrained Patterns, Part 3: Representing Constraint Sets

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

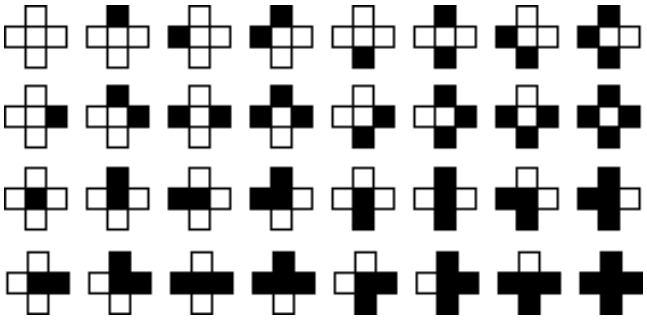
Graphical Representations

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, slightly reduced to fit here, 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

00000000000000011000000000000000

Note that although the cell-bar representation is difficulty 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.

References

1. Griswold, Ralph E. "Constrained Patterns, Part 1: Basic Concepts", 2002: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_con1.pdf

2. Griswold, Ralph E. "Constrained Patterns, Part 2: Neighborhood Analysis", 2002: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_con2.pdf

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