Knitting for fun: a recursive sweater

Anna Bernasconi[1], Chiara Bodei[1], and Linda Pagli[1]

Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo, 3, I-56127, Pisa, Italy. {annab,chiara,pagli}@di.unipi.it

Abstract. In this paper we investigate the relations between knitting and computer science. We show that the two disciplines share many concepts. Computer science, in particular algorithm theory, can suggest a lot of powerful tools that can be used both in descriptive and prescriptive ways and that apparently have not yet been used for creative knitting. The obtained results are short (optimal size) recursive descriptions for complex patterns; creation of new complex recursive patterns; and the application of three-valued algebra operations to combine and create a wide variety of new patterns.

Keywords: Modeling Knitting; Pattern Knitting Diagrams; Checkerboard, Sierpinski, and Butterfly patterns; Knitting Complexity.

1 Knitting, Mathematics and Computer Science

Knitting is usually considered a female activity and females are usually not considered to be inclined to mathematics, or to science in general. Nevertheless mathematical skills are necessary for knitting, because they help to realize symmetries, inversions, scalings and proportions; good abstraction capabilities are indeed needed to figure the final result out and to map the idea of a pattern into a knitted form. Therefore, even illiterate women use mathematics while knitting, without knowing it.

Furthermore, if you think about knitting carefully, you can find a lot of formal and abstract structures. And here, computer science may come in, by providing tools and ways of interpreting and re-interpreting these structures, thus giving a form to knitting.

Knitting offers us a nice chance to revisit some of the main concepts of computer science from a new perspective and, at the same time, knitting can be better understood in the light of this theoretical tour. Computer science, especially algorithm theory, can suggest a lot of powerful tools that can be used both in descriptive and prescriptive ways and that apparently have not yet been used for creative knitting.

A pattern can be seen as a matrix of *stitches* (columns) and *needles* (rows), and it is usually repeated many times horizontally or vertically, or inserted into another pattern or interleaved with one or more other patterns. The stitches can be chosen from a set of possible stitches, but not all the combinations are allowed: we have to select them according to a set of predefined rules, which guarantee a consistent result.

Once a new pattern has been created, its description is represented through the so called *pattern knitting diagram*; in this way the pattern can be reproduced many times and communicated to others. The diagram must be read from the bottom to the top, the odd rows from right to left and the even ones in the opposite direction, i.e., a “bustrophedic” reading (from ancient greek $\beta o \upsilon \varsigma$, "ox" + $\sigma \tau \rho \epsilon \phi \epsilon \iota \nu$, "to turn", imitating the ox ploughing the field, back and forth). Rows and, in general, patterns exhibit structural regularity, which allows us to use the notion of grammars to describe them (Section 2). On the other hand, exactly the bustrophedic reading of the diagram gives the specification, row by row, of the elementary stitches to be performed and essentially represents the *algorithm* to be executed to realize the piece of work.

The relations between mathematics and knitting have been studied from many points of view. A wide review on this subject can be found in the web site *The Home of Mathematical Knitting* [8].

As computer scientists we will mostly consider other aspects of the creative knitting process, which, as we will see, are interesting and, to our knowledge, have not yet been investigated. First of all, recall that one of the first examples of an elementary computer was a mechanical loom, invented by *Joseph Jacquard* in 1801. This machine was able to execute patterns composed of several interleaved threads of different colors following the scheme of punched holes in board punch cards. In this way the Jacquard machine automatically selected the color of each stitch, allowing complex combinations. Nowadays we still use the name jacquard to indicate this kind of pattern and the idea of using punched cards as knitting diagrams to reproduce particular patterns has been used also by more modern knitting machines. Now they include very sophisticated control devices that behave as real dedicated computers and are able to reproduce any complex pattern.

We started our study by asking ourselves what applying recursion to knitting could lead to. First of all we wanted to understand if recursive motives could be employed to obtain beautiful patterns, and then to see how to exploit the power of recursion to create very short descriptions. We found some surprising results as shown by the examples of recursive patterns proposed here, which, in our opinion, show some beauty (Section 4). Moreover, the recursive patterns can be seen as schemes of patterns, from which it is possible, changing the initial conditions and the basic stitches, to obtain families of new patterns, thereby opening a new style of knitting.

In addition, recursive patterns can be defined in a very succinct way, i.e., their pattern knitting diagrams can be automatically generated with very short recursive algorithms. Applying, by analogy, the well known concept of *Kolmogorov Complexity* in this framework, we might say that recursive knitting patterns have low *"knitting complexity"* (Section 5). This result shows how recursion allows us to get an optimal compression of patterns, whose standard description would have a much higher complexity. The usual knitting instructions, described in natural language, can then be derived from the recursive algorithms, or directly from the generated pattern diagrams, in an automatic way by some sort of *knitting compiler*.

A second part of this study is still devoted to creating new patterns, but based on the combination of given patterns. Different combinations of stitches give rise to different textures, among which the most famous are *flat stockinette*, *reverse stockinette*, and *seed stitches* (Section 3). Patterns and motifs are obtained by using different textures for different areas of the knitting piece. Consequently, the elementary unit to be considered seems to be the texture unit, i.e., the stitch processed according to a particular texture. We call this unit *knitting element* or *knittel*, in analogy with pixel (picture element).

We show how considering a set of possible textures and their combinations as a *three-valued algebra*, it is possible to combine shapes and patterns in a very simple and elegant way, using the algebra operations, and to easily obtain the specifications of many nice patterns (Section 6). This is only an example of how a formal approach offers a way to enhance the design possibilities.

2 The Grammar of Knitting

In specialistic journals, patterns are specified both with pattern knitting diagrams and verbal descriptions. A pattern knitting diagram is a matrix, where each element corresponds to a single stitch and every row corresponds to a needle. The pattern can be repeated as many times as needed. Every kind of stitch is represented by a special knitting symbol. In other words, there is a finite alphabet $\mathcal{S}$ of knitting symbols S and a

```
   Cast on over a number of stitches multiple of 4 plus 3.
```

Row 1: `* knit 2, purl 2; repeat from *: to last 3 sts, knit 3.`
`Repeat row 1.`

```
| | | - - | |
| | - - | | |
| | | - - | |
| | - - | | |
| | | - - | |
| | - - | | |
| | | - - | |
```

Fig. 1. *Standard description (top) and pattern knitting diagram (bottom) of Mistake Rib.*

precise syntax of these symbols.

S ::=	*stitches*
$\vert$	knit
$-$	purl
o	cast on
...	

The verbal description is compressed horizontally by inserting the repetitions between two stars, and vertically indicating the rows to be repeated, as in Fig. 1 (see also Fig. 5). The explanation of each row resembles the production rules in a regular grammar [3], where terminals are knitting symbols in $\mathcal{S}$ and one special non terminal symbol suffices to generate each row. Consider, e.g., the first row in Fig. 1. It is easy to rephrase it as the following production (remember that the row should be read from right to left):

$$R \ ::= \ \ ||| \ \mid \ R \ --||$$

that, in turn, generates the following language of words $L(R) = |||\{--||\}^* = \{|||, |||--||, |||--||--||, ...\}$ (see the diagram in Fig. 1, which presents just a single repetition). It is interesting to observe that the pattern descriptions use the star with the same meaning of the Kleene star.

Consequently, grammars appear a good tool for pattern modeling, as they provide simple and elegant representations of patterns. Actually, a pattern corresponds to a two-dimensional word and this calls for a generalization of formal word language theory. There are many possible models for two-dimensional languages that can be used for the knitting

```
- - - -        | | | |        - | - |
| | | |        - - - -        - | - |
- - - -        | | | |        - | - |
| | | |        - - - -        - | - |
```

Fig. 2. *Pattern Knitting Diagrams for Stockinette (left), Reverse Stockinette (center), Seed Stitch (right).*

```
- - - - | | | |    a a a a b b b b
| | | | - - - -    a a a a b b b b
- - - - | | | |    a a a a b b b b
| | | | - - - -    a a a a b b b b
| | | | - - - -    b b b b a a a a
- - - - | | | |    b b b b a a a a
| | | | - - - -    b b b b a a a a
- - - - | | | |    b b b b a a a a
```

Fig. 3. *Checkerboard: Pattern Knitting Diagram (left), Visible Pattern (right).*

framework. This subject is left for future work. The nice thing is that we can define a sweater as a piece of a particular language.

3 Knit Textures

Different combinations of stitches give rise to knit fabrics that result in different textures (see Fig. 2). The basic knit fabric is called *stockinette* pattern and it is obtained by alternating rows of knits with rows of purls on the right side. The visual effect is a grid of V shapes. On the wrong side the pattern has a different texture, the effect is a grid of $\sim$ shapes and it is used as a pattern in itself (obtained by alternating rows of purls with rows of knits) with the name of *reverse stockinette*. Another common fabric is called *seed stitch* and it is obtained by alternating knits and purls. Generally, the visible patterns are not completely congruent to their diagrams. For instance, the visual effect of the seed stitch is that of a checkerboard, while its diagram (the right one in Fig. 2) has a different aspect. This is due to the fact that the odd rows describe the right side of the fabric, whereas even ones refer to the wrong side. When following the diagram, this corresponds to the perspective on the fabric of the person who knits it.

Patterns are usually obtained by combining different textures as in the checkerboard pattern in Fig. 3, where stockinette and reverse stockinette are combined in a checkerboard style in four tiles (see also Section 4). Again, the diagram does not give the immediate intuition of the pattern. To better visualize the pattern, we can use a symbol for the generic point of the stockinette texture, e.g., “a” and another, e.g., the “b” for the generic point of the reverse stockinette. Sometimes a similar convention is used in pattern knitting diagrams. In other words, we are using an abstraction of the texture, namely the *knitting element* or *knittel*, the elementary texture unit. In our example, the knittel symbol “a” stands for the generic point of the stockinette, that can be either a knit or a purl, depending on the position in the stockinette area. Of course, we can also associate “a” and “b” to other pairs of textures and obtain new combinations. For ease of presentation, in Section 6 we will use a color code for textures, associating them to different scales of gray.

4 Recursive knitting: an algorithmic description

How can we apply the powerful concept of “recursion” to the knitting world? Probably, the most natural way is that of exploiting the recursion in the description of the knitting patterns, by using recursive algorithms to automatically generate them. One additional advantage of our algorithmic description over standard ones based on instructions in natural languages as well as pattern knitting diagrams, is that with just one algorithm it is possible to obtain whole families of new patterns by simply changing the initial conditions and the basic stitches.

We propose here three examples of families of patterns of increasing *difficulty*, as well as *beauty*, that can be generated recursively. The first example concerns a pattern that could be easily defined in an iterative way; in the second example we consider a pattern based on a well known plane fractal; finally, the pattern shown in the third example is nothing other than a knitted *butterfly network*.

Without loss of generality, we consider the generation of square knitting diagrams, each represented as a matrix a, whose entries describe single stitches. The generation of the knitting diagram is performed in two steps: first the execution of a recursive algorithm generates the pattern, which resembles its final aspect, and then the associated knitting diagram is simply produced by inverting every other row (i.e., changing the knit stitches into purl ones, and viceversa).

Checkerboard pattern. As said above, a Checkerboard pattern is composed of identical squares that alternate between stockinette stitch and reverse stockinette stitch. The only variable is the dimension of the squares, i.e., the number of stitches across and rows long. This pattern can be seen as a generalization of the well known *seed stitch*, obtained when the dimension of the squares is one. The input of the algorithm is given by

```
| | | | - - - -     | | - - | | - -     | - | - | - | -
| | | | - - - -     | | - - | | - -     - | - | - | - |
| | | | - - - -     - - | | - - | |     | - | - | - | -
| | | | - - - -     - - | | - - | |     - | - | - | - |
- - - - | | | |     | | - - | | - -     | - | - | - | -
- - - - | | | |     | | - - | | - -     - | - | - | - |
- - - - | | | |     - - | | - - | |     | - | - | - | -
- - - - | | | |     - - | | - - | |     - | - | - | - |

- - - - | | | |     - - | | - - | |     - | - | - | - |
| | | | - - - -     | | - - | | - -     - | - | - | - |
- - - - | | | |     | | - - | | - -     - | - | - | - |
| | | | - - - -     - - | | - - | |     - | - | - | - |
| | | | - - - -     - - | | - - | |     - | - | - | - |
- - - - | | | |     | | - - | | - -     - | - | - | - |
| | | | - - - -     | | - - | | - -     - | - | - | - |
- - - - | | | |     - - | | - - | |     - | - | - | - |
```

Fig. 4. *Checkerboard visible patterns (top) and their knitting diagrams (bottom), with increasing resolution. In the visible patterns the symbol* | (-) *stands for a knittel of a (reverse) stockinette fabric.*

a matrix a, whose entries are initialized as knit stitches, its dimension $n = 2^k$, the dimension d of the basic pattern, and the indexes x and y used to indicate the portion of matrix to fill.

```
**Checkerboard**$(a, n, d, x, y)$
    **if** $(n == d)$
        **for** $(i = 0;\ i < d/2;\ i{+}{+})$
            **for** $(j = 0;\ j < d/2;\ j{+}{+})$
                $a[i{+}d/2{+}x][j{+}y] = a[i{+}x][j{+}d/2{+}y] = -$ ;
    **else**
        **for** $(k = 0;\ k < 4;\ k{+}{+})$
            $i = k/2;$
            $j = k \bmod 2;$
            Checkerboard$(a, n/2, d, x + i * n/2, y + j * n/2);$
```

Observe that by changing the value of d from its maximum value n down to its minimum value 2, we obtain patterns with progressively increasing resolution. Examples of patterns of the Checkerboard family and of their corresponding knitting diagrams are shown in Fig. 4. Finally, Fig. 5 shows a standard description of a Checkerboard pattern of resolution 2, by instruction in natural languages.

Sierpinski pattern. The definition of this pattern is based on the plane fractal known as *Sierpinski carpet*, first described by Wacław Sierpiński in 1916. The construction of the Sierpinski carpet begins with a square. The square is cut into 9 congruent subsquares in a 3-by-3 grid, and the central

```
Cast on over a number n of stitches, multiple of 4.
  Row 1: * knit 2, purl 2; repeat from *.
  Row 2: repeat row 1.
  Row 3: * purl 2, knit 2; repeat from *.
  Row 4: repeat row 2.
  Repeat rows 1, 2, 3, and 4 for n/4 times.
```

Fig. 5. *Standard description of a Checkerboard pattern for a fixed value $d = 2$.*

subsquare is *removed.* The same procedure is then applied recursively to the remaining 8 subsquares, depending on the chosen resolution.

To realize such a pattern with our tools, i.e., needles and handwork, we only have to decide how to "remove" a square. This could be done, e.g., by using stockinette stitch as background, and reverse stockinette stitch for the removed squares, or viceversa. As before, the input of the algorithm is given by a matrix a, whose entries are initialized as knit stitches, its dimension $n = 3^k$, the dimension d of the basic pattern, and the indexes x and y indicating the portion of matrix to fill.

```
Sierpinski($a, n, d, x, y$)
    if ($n == d$)
        for ($i = d/3$; $i < 2 * d/3$; $i$++)
            for ($j = d/3$; $j < 2 * d/3$; $j$++)
                $a[i+x][j+y] = -$ ;
    else
        for ($i = n/3$; $i < 2 * n/3$; $i$++)
            for ($j = n/3$; $j < 2 * n/3$; $j$++)
                $a[i+x][j+y] = -$ ;
        for ($k = 0$; $k < 9$; $k$++)
            if ($k \neq 4$)
                $i = k/3$;
                $j = k \bmod 3$;
                Sierpinski($a, n/3, d, x + i * n/3, y + j * n/3$);
```

Again, decreasing d from n down to 3, we obtain patterns of progressively increasing resolution. Examples of patterns of the Sierpinski family, together with their knitting diagrams, are shown in Fig. 6. Whereas in the previous example we were able to give a concise standard description for a given resolution by verbal knitting instructions, for the present pattern a similar concision could not be attained.

Butterfly pattern. Our last example of recursive knitting pattern is based on the well known notion of *butterfly network* (see [5]):

Definition 1. *A d-dimensional butterfly has $(d + 1)2^d$ nodes and $d\,2^{d+1}$ edges. The nodes correspond to pairs (w, i) where i is the level of the node*

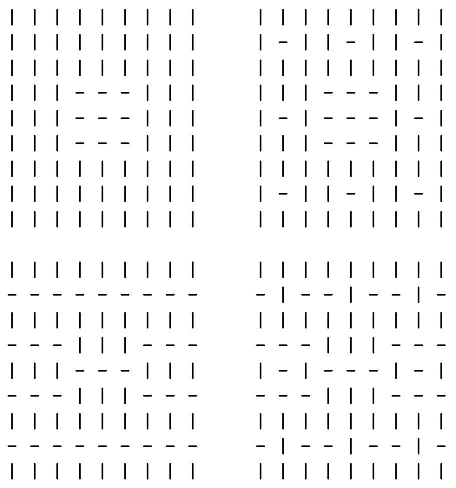

Fig. 6. *Sierpinski visible patterns (top) and their knitting diagrams (bottom), with increasing resolution.*

$(0 \leq i \leq d)$ *and w is a d-bit binary number that denotes the row of the node. Two nodes (w, i) and (w', i') are linked by an edge if and only if $i' = i + 1$ and either w and w' are identical* (straight edge) *or they differ in precisely the i'th bit* (cross edge).

This time, a recursive algorithm, in a classical à la *divide et impera* style, is used to generate a scheme of a d-dimensional butterfly (see Fig. 7). For aesthetic reasons, we only consider cross edges. Starting from such a scheme, one can easily derive a matrix describing the visible pattern, and then build the associated knitting diagram. Observe that as in the previous example, yet even more so, a request for brief standard knitting instructions cannot be satisfied for the present pattern.

The input of the algorithm is given by a matrix a, its dimension $n = 2^{d+1} - 2$, and the index x used to indicate the portion of matrix to fill.

```
**Butterfly**$(a, n, x)$
    $m = (n + 2)/2$;
    **if** $(m == 2)$
        $a[0][x] = a[1][x{+}1] = \backslash$;
        $a[0][x{+}1] = a[1][x] = /$;
    **else**
        Butterfly$(a, m - 2, x)$;
        Butterfly$(a, m - 2, x + 1 + n/2)$;
        Combine$(a, m, x)$;
```

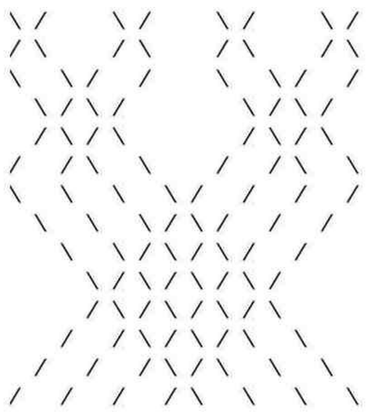

Fig. 7. *A three-dimensional butterfly scheme recursively generated.*

```
**Combine**$(a, m, x)$
    $r = m - 2$;
    **for** $(d = 0;\ d < m;\ d = d + 2)$
        **for** $(j = r;\ j < 2 * m - 2;\ j{+}{+})$
            $a[j][d{+}j{-}r{+}x] = \backslash$;
    **for** $(d = m - 1;\ d < 2 * m - 2;\ d = d + 2)$
        **for** $(j = r;\ j < 2 * m - 2;\ j{+}{+})$
            $a[j][d{-}j{+}r{+}x] = /$;
```

The Butterfly pattern can be realized with needles in various way. For instance, one could play with the two basic stitches, knit and purl, and use them to realize the diagonal lines representing its edges, or, even better, the cable stitch, as shown in Fig. 8.

5 Knitting complexity

In computer science, the *Kolmogorov complexity* (also known as algorithmic entropy, or program-size complexity) of an object is a measure of the computational resources needed to specify the object [6]. For the world of knitting, by analogy we introduce the following:

Definition 2. *The* knitting complexity *of a knitting pattern is the length in bits, expressed in order of magnitude, of the shortest description of its knitting diagram.*

We pose a basic proposition providing an immediate lower bound.

Proposition 1. *A knitting pattern of dimension $n \times m$ has a knitting complexity $\Omega(\log n + \log m)$.*

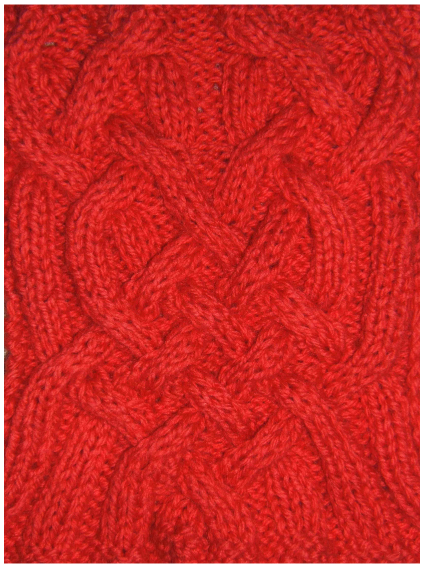

Fig. 8. *Cable stitch realization of the Butterfly pattern.*

Proof. The bound easily follows by noting that $\log n + \log m$ bits are required to specify the diagram size.

Observe that the more a pattern is elementary, the more its diagram is repetitive and easy to describe with "concise" instructions. On the other hand, if a pattern does not present any structural regularity, the shortest description of its knitting diagram will consist of the diagram itself, thus requiring $\Theta(nm)$ bits.

We will now analyse the knitting complexity of the three examples described in Section 4. We have

Proposition 2. *The Checkerboard, Sierpinski and Butterfly patterns have knitting complexity $\Theta(\log n)$, therefore their recursive descriptions are optimal.*

Proof. First of all observe that in this case $m = n$. The proof is constructively obtained from the algorithmic description of these three patterns, specified in Section 4. In fact, each algorithm consists of a constant number of instructions, and the values of the variables and input parameters are all upper bounded by n. Therefore $\Theta(\log n)$ bits are sufficient to describe them. The optimality immediately follows from Proposition 1.

Observe that the optimality has been obtained thanks to the power of the recursive description of the patterns. It can be easily seen that using

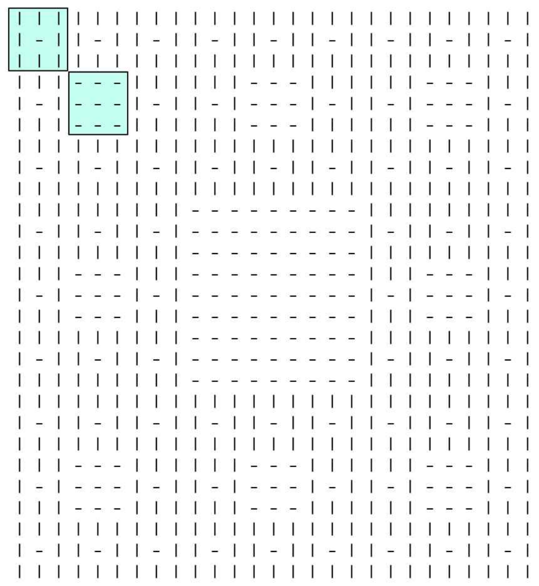

Fig. 9. *A Sierpinsky visible pattern of size* $n = 27$ *and* $d = 3$.

the standard knitting description techniques (natural language or pattern knitting diagrams) only the complexity of the Checkerboard pattern would be of order $\Theta(\log n)$, since it takes a constant number of instructions to be described (see Fig. 5). For the Sierpinski pattern, an optimal compression can be easily obtained when the resolution is very low, e.g., $d = n$. For higher resolution ($d < n$), we can observe that the whole visible pattern can be obtained by the composition of only two basic $d \times d$ subsquares, as those shown in Fig. 9. For any d, these two squares can be described with $\Theta(\log d)$ bits, because of their regularity. The overall diagram can then be described as an $n/d \times n/d$ array of such subsquares. In this way we obtain a description of size $\Theta(n^2/d^2 + \log d)$.

Using the standard knitting description techniques, for the Butterfly pattern we are not able to find a better description than the whole array of stitches, requiring $\Theta(n^2)$ bits.

6 Designing Patterns using Algebra

Patterns and motifs are obtained by using different textures for different areas of the knitting piece. Previous studies in this direction can be found in [2, 7]. For ease of presentation, suppose we only have three different kinds of texture in the same pattern. To obtain new patterns, we play with a three-valued algebra, i.e., an algebra over the finite field $GF(3)$. $GF(3)$ contains three elements, usually labelled with 0, 1 and 2, and arithmetic is performed modulo 3. As operations, we mainly focus on the sum and multiplication modulo 3 (typical of this ring). Note that the sum and multiplication modulo 2, i.e. the logical exclusive or (XOR) and AND operations, and the other boolean operations can be obtained as combinations of the typical $GF(3)$ operations.

We apply element-wise these operations to matrices of knittels, assuming values in $\{0, 1, 2\}$, depending on the corresponding texture. We will take two patterns, suitably coded in our algebra, and apply one or more operations. We can also combine more than two patterns. We use a color code for each value in $\{0, 1, 2\}$, by associating them to different scales of gray, to help in visualizing the patterns (0.01 for 0, 0.5 for 1, and 0.95 for 2).

Boolean operations are usually used to combine patterns and shapes in computer vision. Resorting to a third value increases the number of possible combinations. The role of values is twofold: on the one hand we can associate each value to a different texture and obtain three textures; on the other hand, their combination with different operations can play a role in the manipulation of patterns. We can think about a generalization of the masking notion, to manipulate knittel areas in bulk. It suffices to use the second pattern as a mask for the first one. If the operation applied is the multiplication modulo 3, we can obtain the following effects on the first pattern: (i) if the mask area consists of 0 the effect on the corresponding area in the first pattern is that of clearing, i.e., the area it is cleared to zero regardless of the initial value; (ii) if the mask area consists of 1, the original values in the corresponding area are not changed, while (iii) if the mask area consists of 2, the values 1 and 2 are inverted. Similar considerations can be made for the sum modulo 3, and for all the operations possible in $GF(3)$.

We present two experiments, in which we consider $n \times n$ square matrices. In the first example, in Fig. 10, we can observe the effect of the multiplication modulo 3, depending on the values.

An interesting variation could be to apply different operations to different sub-matrices, i.e., having also matrices of operations, as in Fig. 11, where each operation is applied to each $n/2 \times n/2$ sub-matrix. The matrix of operations *comb* is the following: the effect is nicely kaleidoscopic.

$$\begin{bmatrix} +_{mod\,3} & \times_{mod\,3} \\ \times_{mod\,3} & +_{mod\,3} \end{bmatrix}$$

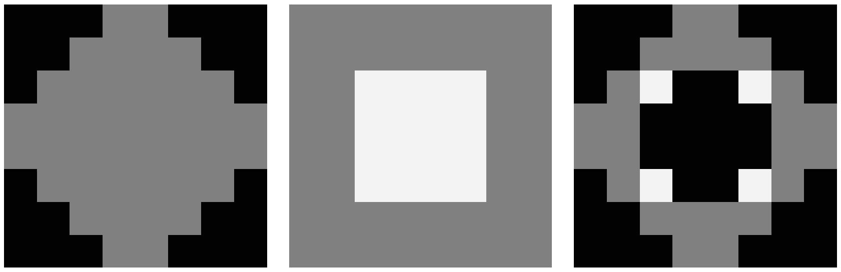

Fig. 10. *Multiplication modulo 3 (right) of the two patterns A (left) and B (center).*

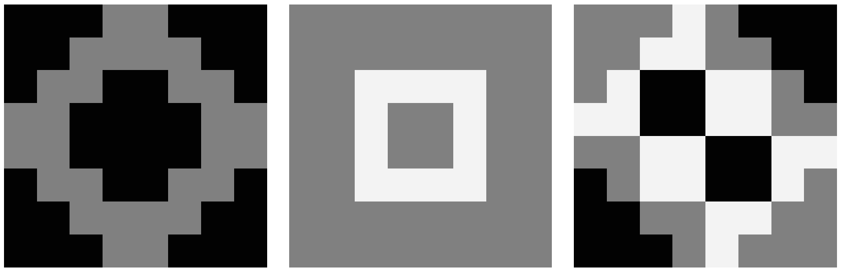

Fig. 11. *Combination (right) of the two patterns C (left) and D (center) with the matrix of operations comb.*

7 Concluding remarks

We conclude here our tour of the computational aspects of the knitting world. Summarizing, the results we have obtained are:

- Short (optimal size) recursive description for complex patterns.
- Creation of new complex patterns.
- Application of three-valued algebra operations to combine and create a wide variety of new patterns.

Beside the theoretical interest, the above results have also practical impact. In fact, using a deep level of recursion and high resolution, we can obtain automatically and in a very simple way, arbitrarily complex patterns, never designed nor produced before, to our knowledge. Their pattern knitting diagrams can be also obtained in an automatic way. Note that, even if a very complex pattern will probably require a greater skill or concentration in the executor, if human, knitting is a sequential process where one stitch is processed after the other, and therefore the overall processing time remains linear in the size of the array. A complex pattern can be obtained in approximately the same time as a simple one. This is true in particular for knitting machines whose execution time, following a program, is independent of the difficulty of the diagram.

The exploration of the knitting world with the eye of the computer scientist opens a variety of interesting topics beside those considered: this paper is only the starting point for further investigation.

It could be also nice to organize, for educational purposes, an introduction to basic concepts of computer science completely based on knitting, obviously for women only!

Acknowledgments. We would like to thank the numerous men who, with a mixture of diffidence and curiosity, helped us with this paper.

References

1. P. Allen, T. Malcolm, R. Tennant, and C. Fall, *Knitting for Dummies*, 2002.
2. R.E. Griswold, *Designing Weave Structures Using Boolean Operations, Part 1, 2, 3*, `http://www.cs.arizona.edu/patterns/weaving/webdocs.html`
3. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman *Introduction to Automata Theory, Languages, and Computation* Addison-Wesley, 2001.
4. D.E. Knuth, *The Art of Computer Algorithms: Sorting and Searching*, Vol. 3, Addison-Wesley, Reading, MA, 1973.
5. F.T. Leighton, *Parallel Algorithms and Architectures: Array, Trees, Hypercubes*, Morgan Kaufmann Publishers, San Mateo, CA, 1992.
6. M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer Verlag, New York, 2005.
7. D. Suzuki, T. Miyazaki, K. Yamada, T. Nakamura, and H. Itoh, *A Supporting System for Colored Knitting Design*, Proc. of the 13th Int. Conf. on Industrial and engineering applications of artificial intelligence and expert systems, IEA/AIE, Springer-Verlag New York, 2000.
8. *The Home of Mathematical Knitting* `http://www.toroidalsnark.net/mathknit.html`