**The Stranded Cellular Automata Model**

Cellular automata are mathematical models that represent an initial condition changing over time. As the name implies, they consist of cells with states that are “neighbors” to each other and change their states based on the states of their neighbors. In the case of the Stranded Cellular Automata(SCA) created by Dr. Holden, each cell has 8 possible states and 2 neighbor cells

\*wide graphic showing all 8 states and a group of cells circled showing what cells are neighbors to what cells\*

In order to distinguish the two types of crossings, we will refer to the crossing with the strand on top running like the slant in the letter Z as a “z-cross” and the opposite crossing with the strand on top running like the slant in the letter S as a “s-cross”

\*graphic showing the difference between z-crosses and s-crosses\*

The calculation of each cell’s state based on its neighbor pair is split into two different rules: the “turning rule”, which governs whether or not strands will slant/cross, and the “crossing rule”, which dictates which strand goes over the other in the case of a cross. Each of these rules are broken down into 9 binary bits that cover the 9 cases of neighbor pairs. Since each of these bits is labeled 0-8, it is possible to write out each rule in decimal notation. Example: Instead of writing turning rule 101000100, it is more concise to write turning rule 324 (the equivalent base 10 number)

\*include both table sample grid graphics, with labels for each bit number\*

It is important to distinguish that the neighbor pair cells do not have to exactly match the pairings shown in figure X/Y. For the turning rule, cells with double straight strands can be substituted in pairings that have single straight strands, and cells with a single strand that slants away from the new cell’s inputs can be substituted in pairings that have cells with no strands.

\*graphic showing turning rule equivalent cells\*

**Representing Braids with Stranded Cellular Automata**

We can use Stranded Cellular Automata to model various types of braids with different numbers of strands. Braids, unlike weaves, have finite width because they reuse the same strands. This means that there is no need to let the border cells “wrap around” as Hao Yang defined them in his work with weaves. In a similar vein, our turning rule for representing braids will not be fixed due to the nature of braids containing both slanted and upright parts.

We started off by constructing physical models of the braids to analyze. We then transcribed the crossings and strands as their corresponding cell states in a Stranded Cellular Automata. Upon checking the output of each neighbor pairing, we were able to derive an initial condition, turning rule, and crossing rule that generated a braid identical to the model.

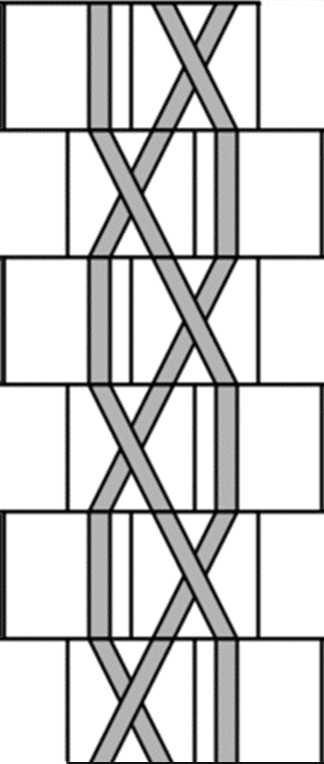
 

Figure 1: 3-Strand Braid and its SCA counterpart (Turning Rule 68, Crossing Rule 32)

After analyzing the simple 3-strand braid and finding no issues with converting it into an SCA, we decided to add another strand to add to the complexity. We found two 4-strand braids that were representable by SCA, a “flat” and “square” pair of braids that both used the same turning rule but different crossing rules.

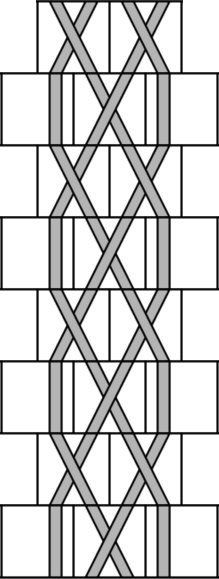
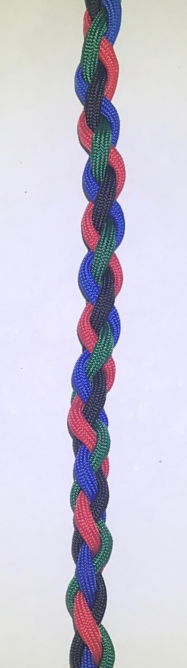
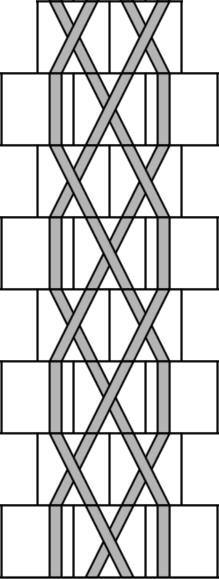
   

Figure 2: Flat 4-Strand Braid with SCA counterpart Figure 3: Square 4-Strand Braid with SCA counterpart

(Turning Rule 324, Crossing Rule 4) (Turning Rule 324, Crossing Rule 140)

An interesting observation made when comparing 3-strand braids to 4-strand braids was the “backwards compatibility” of the turning rule shared by the two 4-strand braids we analyzed.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Number | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Decimal |
| 3-Strand Turning Rule | 0 | **0** | **1** | **0** | 0 | 0 | **1** | 0 | 0 | 68 |
| 4-Strand Turning Rule | **1** | **0** | **1** | **0** | 0 | 0 | **1** | 0 | 0 | 324 |

Since the case that bit 8 governs in the turning rule does not appear in the 3-strand braid, the value of bit 8 is irrelevant in choosing a turning rule to represent the 3-strand braid. Therefore, it is possible to reuse the turning rule from the 4-strand braids to generate a 3-strand braid identical to the original. However, the case that bit 8 governs in the turning rule does appear in both 4-strand braids so the turning rule of the 3-strand braid would not generate the same braids.

For the case of braids with 5 strands, there was a lot more room for experimentation as different combinations of cells that could not be represented with 3 or 4 strands became available. To start, we took the idea of the backwards compatibility of the turning rule 324 and used it to prototype new braids by varying the crossings. The result of this was a braid whose generations alternated between having 2 Z-crosses and 2 S-crosses. Because each generation contained 2 slanting strands that alternated every generation, we referred to it as the “double slant” braid.

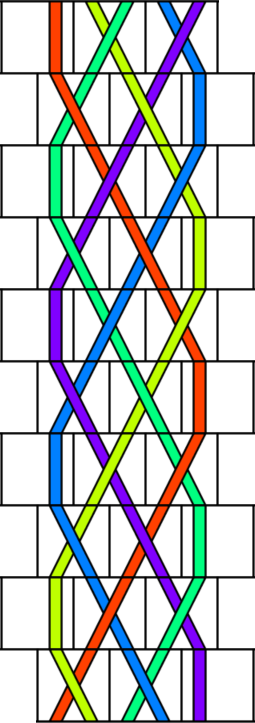


Figure 4: Double slant 5-strand braid with SCA counterpart

* V-shaped 5-strand
* Over under 3+2 braid
* Over only 3+2 braid