Analysis of the Turing State Machine

Lisa Piccirillo - Solving the Conway Knot

https://en.wikipedia.org/wiki/Lisa_Piccirillo

Matt Parker - Stand-Up Maths

http://standupmaths.com/

Hypercube and Turing Machine Videos

Moritz Firsching - Unfolding of the Hypercube

https://unfolding.apperceptual.com/

Pascal Michel - Busy Beaver Game (Explanation)

https://webusers.imj-prg.fr/~pascal.michel/

The inspiration for this paper came from thinking about why the Turing State Machine gives such interesting patterns for tree diagrams and how we could look at the state machine itself without data input to gain a better understanding of the mechanisms at work.

This paper follows an “Analysis of Hypercube Tree Topology” and “Analysis of Conway - Piccirillo Knot with Turing State Machine”. Both papers that are a recommended read prior to this as they are the thought process path that led to this paper.

We seek to analyse sigma(0,0) in this paper using the same logic applied with the below two papers.

https://github.com/Sandcrawler/hypercube

https://github.com/Sandcrawler/knots

Version

Version	
v1	First Draft
v2	Updated diagrams based on observation in analysis of matrice multiplication. (array pointer tracks are not combined). Figure 7.1, Figure 7.2, Figure 8.1, Figure 9.1, Figure 9.2

Sigma(0,0)

The idea of sigma(0,0) may be harder to explain in the first instance than the resultant patterns that can be produced from looking at what it does.

I will attempt in the most simple and logical way possible to strip back a Turing State Machine to sigma(0,0) and attempt to run it with the logic that it has.

Turing Logic:

Read Array

Write to Array

Shift Array

Go to State (HALT if instructed immediately)

This state machine is controlled by a simple set of instructions.

State - a machine state

Symbol - a symbol to write

Direction - a direction to shift

Halt - when to stop

Typically an instruction set will take some form that looks like this matrix.

Where instruction A0 would follow:

Read the array

Write a 1 to the array

Shift the array left

Go to C(<INSERT Read symbol from array here>)

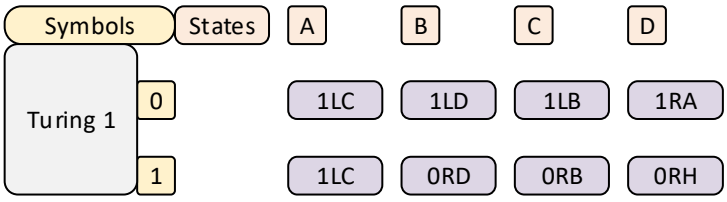

Figure 1.1

The above represents a sigma(4,2).

Sigma(0,0) Representation

Below is the best representation that can be given.

The state is NULL.

The symbol is NULL.

We have no symbol to insert into the instruction set and so are left with LH (Left HALT)

We have no array pointer.

We have no array initialised to 0.

But we have array shift since the state and symbol being NULL do not detract from there being a left or a right instruction being issued due to the ruleset of the Turing State Machine.

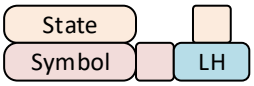

Figure 2.1

We build out a full instruction set.

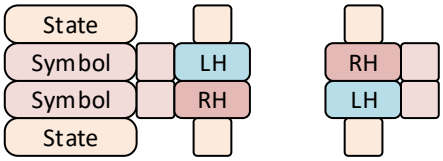

Figure 2.2

As in previous papers we have the inverse instruction set (as it the same answer effectively).

Running the machine

This is the same as any instruction set. Follow the rules to run the machine.

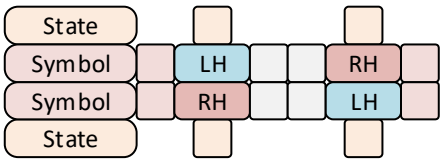

Figure 3.1

We run the machine with NULL.

Read NULL

Write NULL

Left shift (right shift on inverse)

HALT

The grey box shows what I can only describe as spin or NULL density, for lack of a better term.

This one rotation does not show a lot of information yet. However, if we start to build more of this pattern and consider various combining methods from previous papers and apply the same logic over the resultant information.

A larger set

We see here in the larger set that the combined track occur when building out multiple rows of similar objects.

As in the previous papers referenced at the start, we observed that building additional patterns parallel to each other offset allows for an amplified combination to occur.

Here we see that when building out to the X axis further, the offset naturally occurs within the combined purple locations.

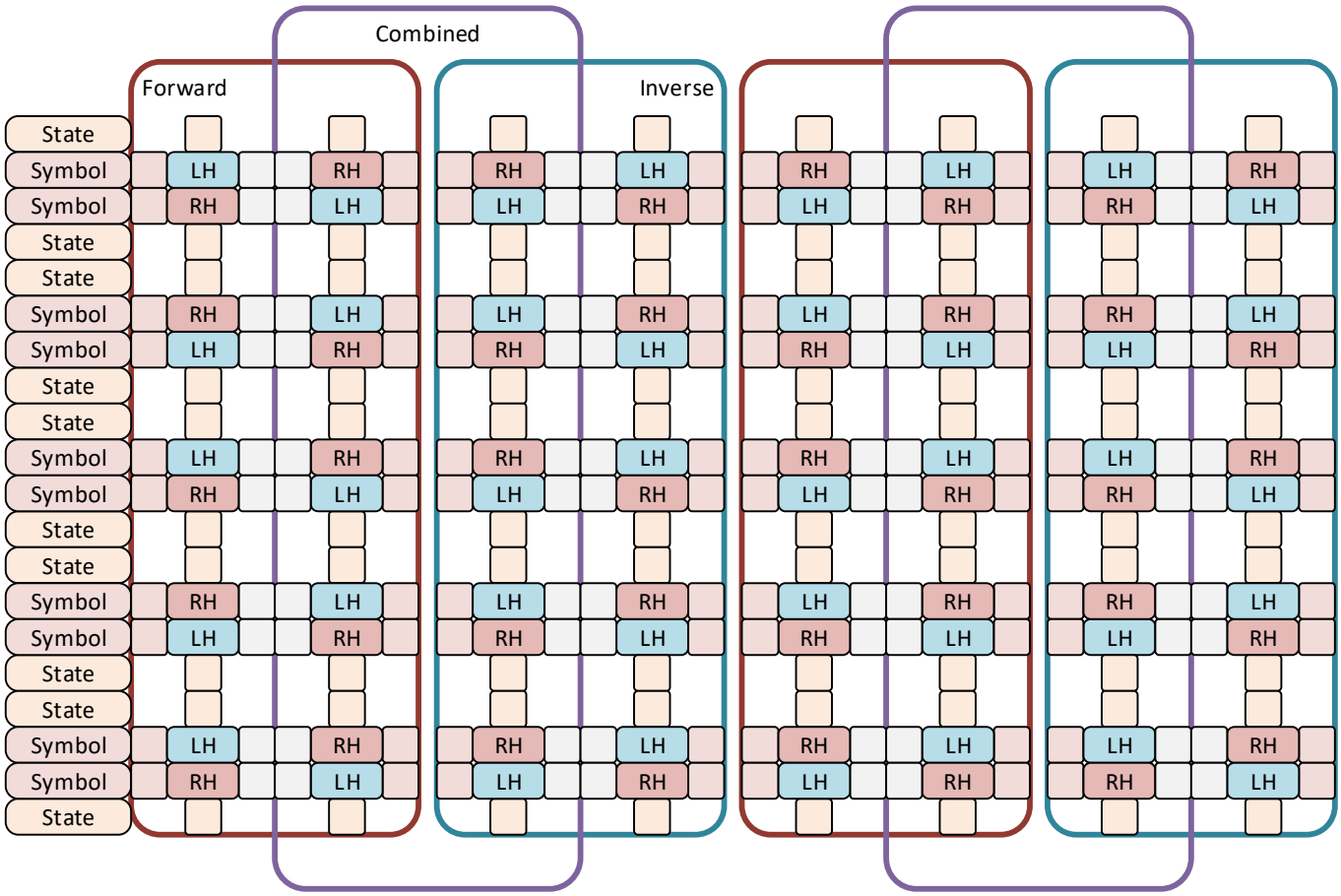

Figure 4.1

Step 1 - Combining Similar State Machines

We rotate the states and the NULL density to face each other.

This is primarily to show in the diagram how the combination occurs so as we can see the pattern better. However, there are some mechanisms at work here by doing this in itself, simply rotating the idea of a spin density and a NULL state flag around the instruction itself.

We can show this idea by bisecting the diagram below.

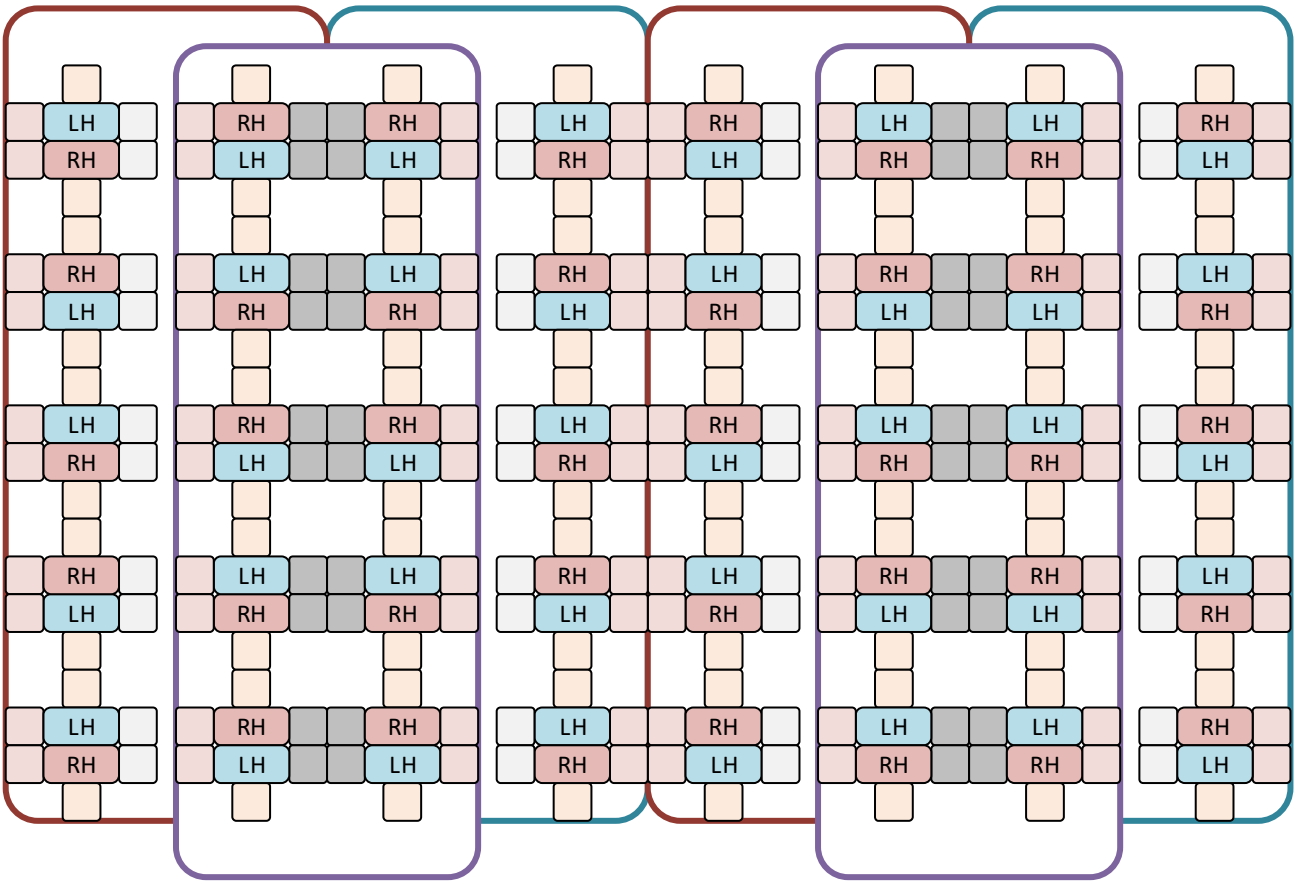

Figure 5.1

Bisecting Step 1

We see in this diagram a slice through the state machine. The black box will be us looking up (along the Y axis of the paper) and the grey box will be where we look down (again through the Y axis of the paper).

The bisection aims to show what happens inside the machine at the point immediately after running the instructions.

One could bisect at each step before during and after to see and compare, this seemed the most interesting step as it shows the result of an action - even if it is just us moving the pattern on the paper.

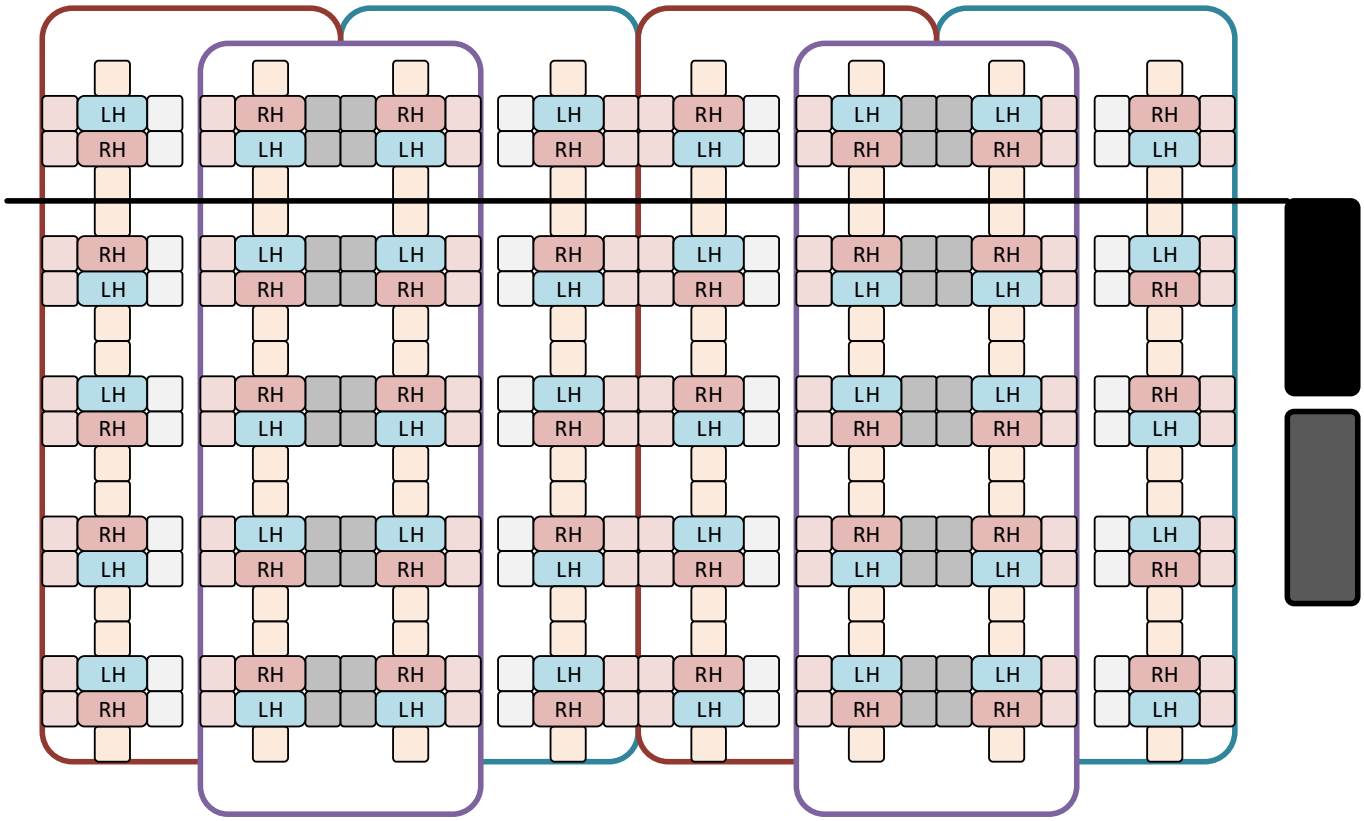

Figure 6.1

The resultant bisection of the Turing State Machine, looking up and looking down, respectively.

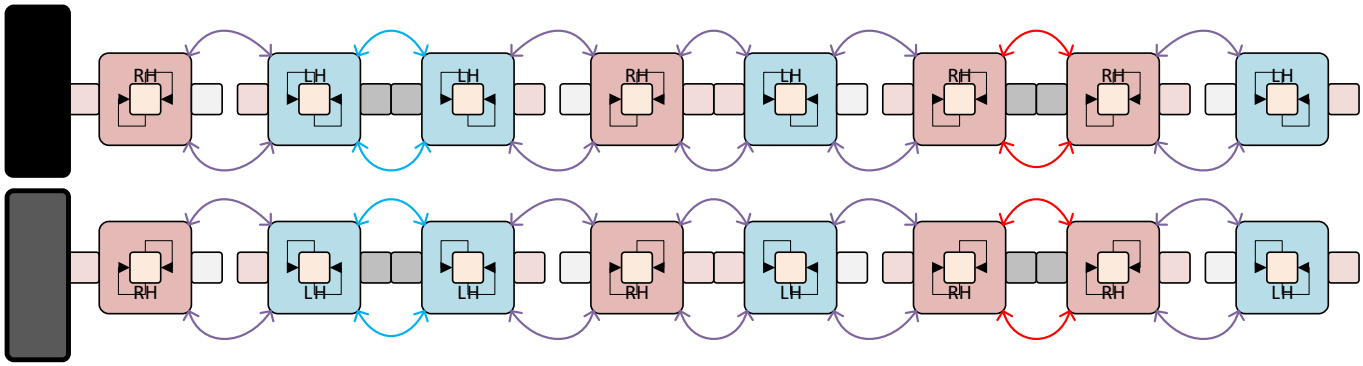

Figure 6.2

Step 2 - Representation of Data

We have two options for how to represent the data. Either blocks or stitching.

Blocking

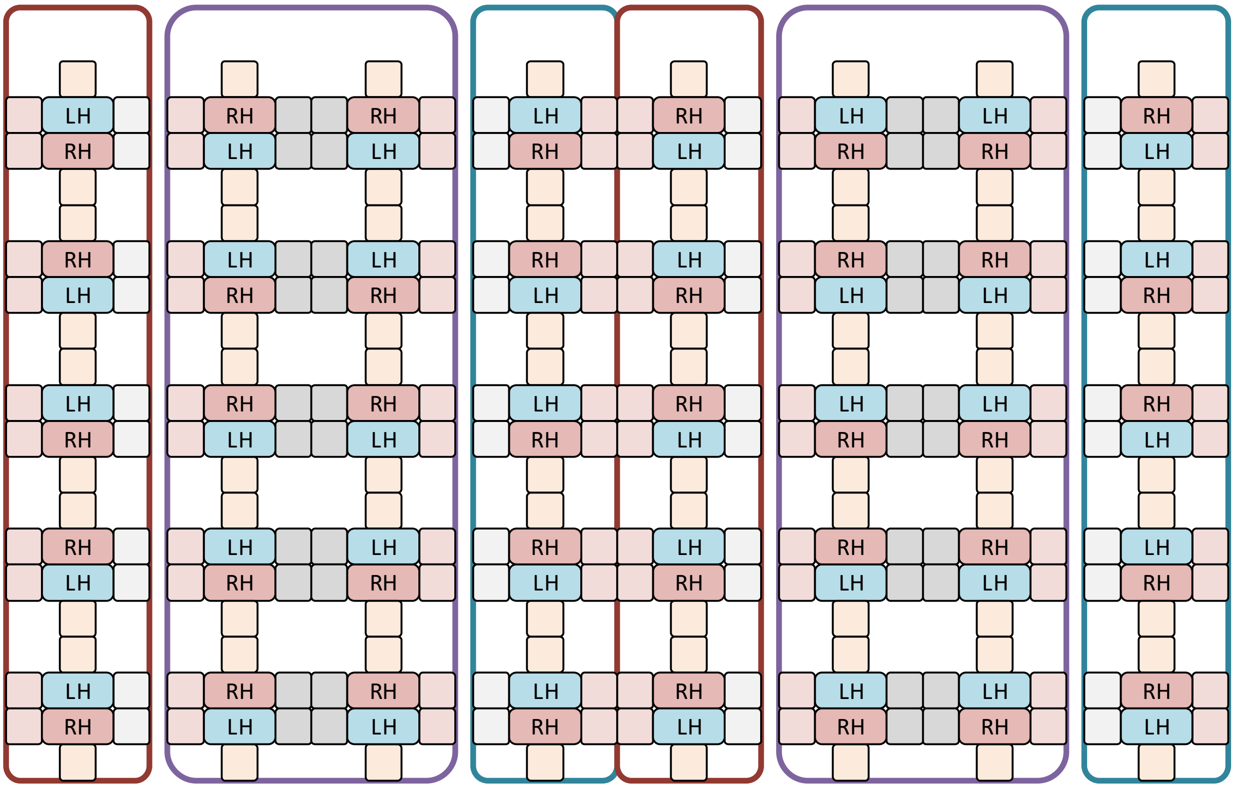

Figure 7.1

Stitching

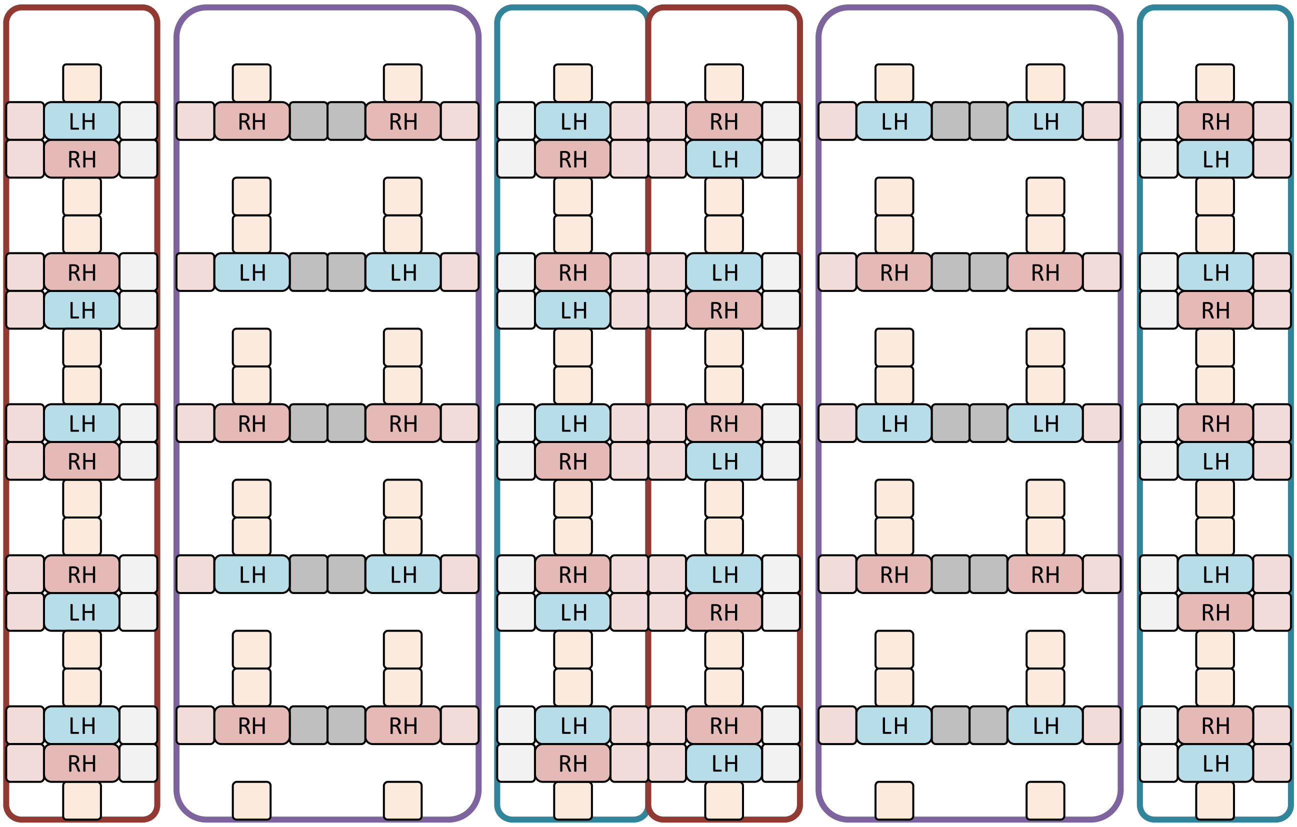

Figure 7.2

Step 3 - Block Horizontal Combination

Here we can see a familiar pattern arise in the horizontal plane we see matching spin pairs across the Y plane.

If we were to continue with this block pattern it is possible that by alternating the collapse horizontally and then vertically a ripple like pattern would occur across all the tracks, with the idea of spin density increasing in a lattice like pattern across folds.

The diagrams in next section show this same pattern without the states and symbols for a clearer view.

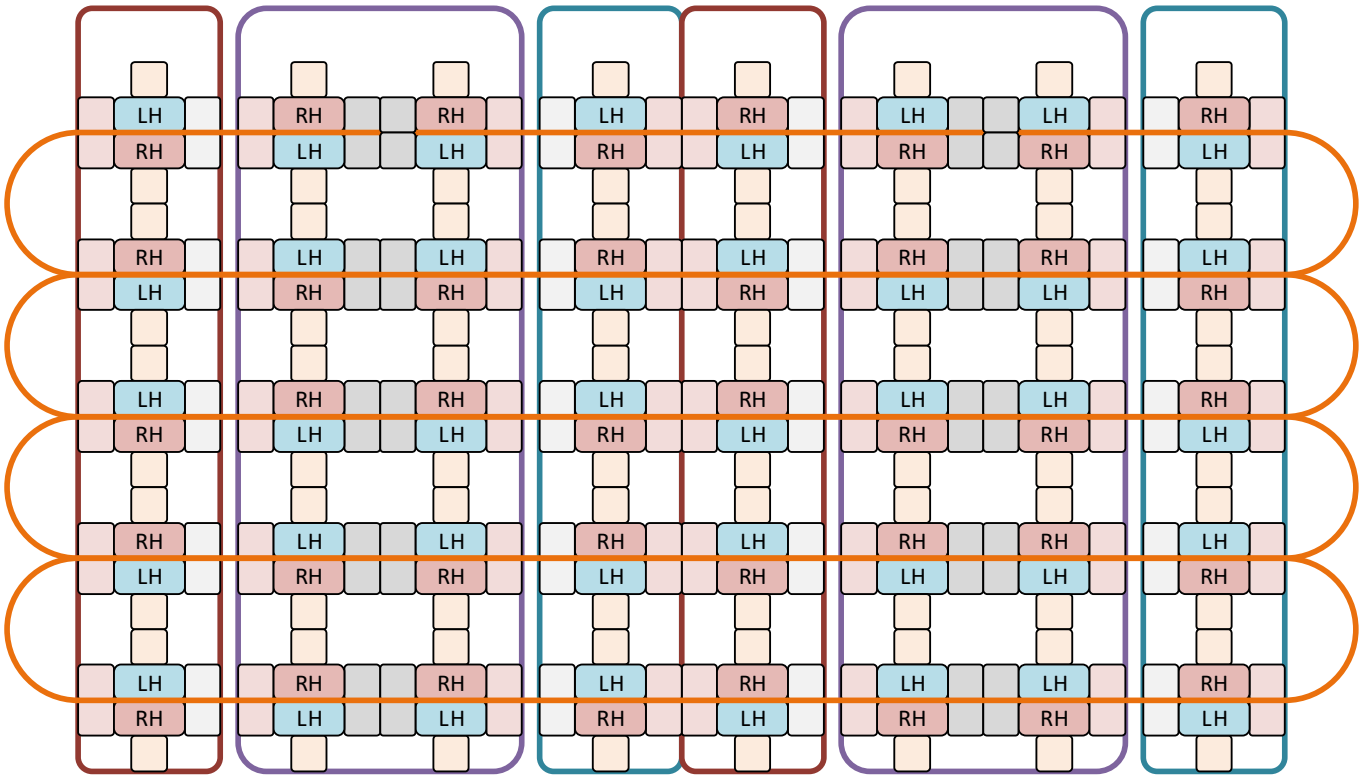

Figure 8.1

Step 4 - Condensed Representation

We condense by removing what is NULL. States and Symbols are both removed from the below diagrams. The diagrams detail the density of the spin.

Blocks

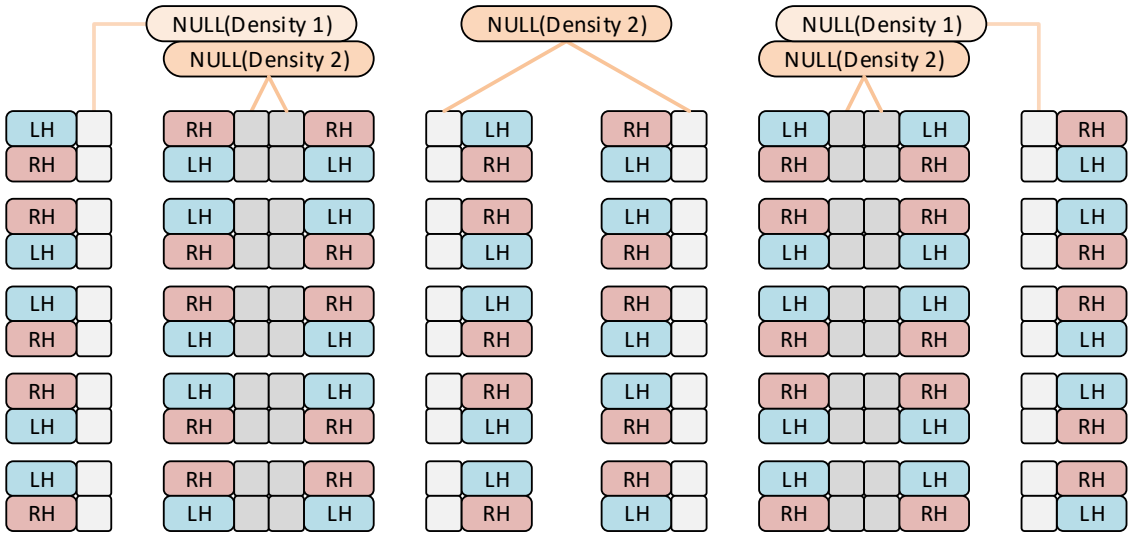

Figure 9.1

It is possible to collapse this pattern both vertically and horizontally.

Stitching

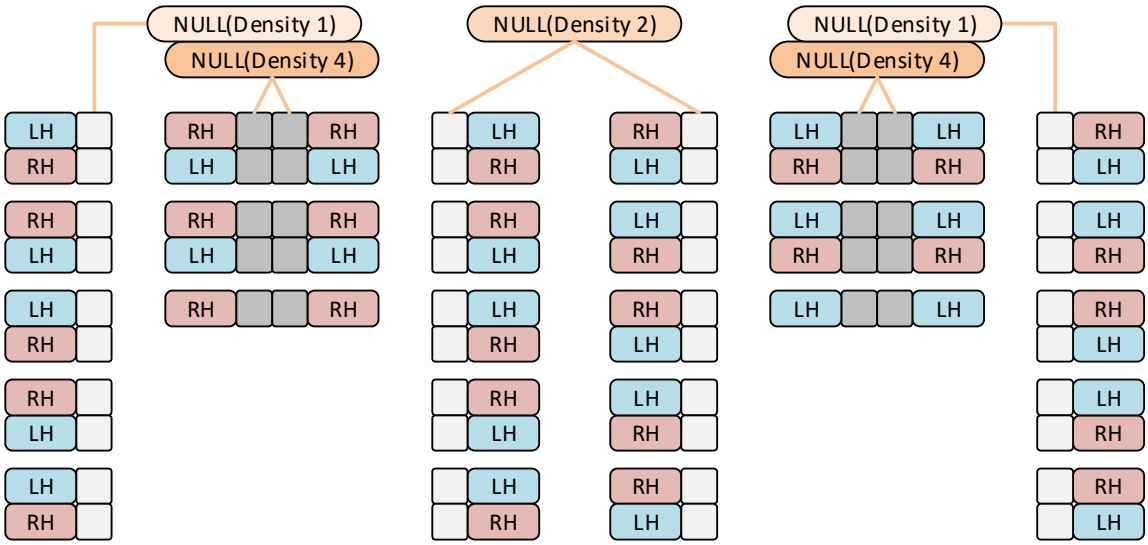

Figure 9.2

It is not possible to collapse this pattern both vertically and horizontally.

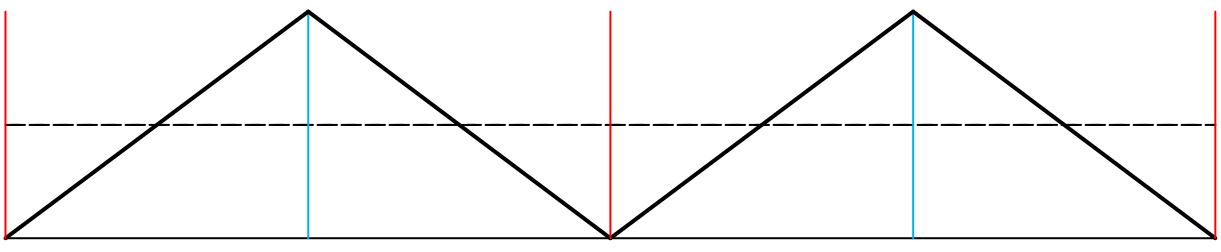

Figure 9.3

A representation of the shortening of the left-right oscillating pattern when collapsing to stitched state due to less rows.

Observations

Observation #1

The idea of spin or NULL density (the equivalent of shifting an un-initialised array but giving it some notation - ~~the colours grey through black~~ shades of grey in this papers case and assigned a NULL density value for further abstraction).

Observation #2

The idea of actioning an array shift without any input.

There is no idea of initial state, there is no state.

There is no idea of symbol to initialise the array with and so we have an empty array(?).

We still have an array pointer head (that is part of the machine(?)).

Observation #3

We do not shift the states or symbols. We do not shift the array pointer head.

We shift the array.

If there is no initialised value in the array, the array still exists so we can still shift the array (is the array part of the machine initialised or not(?)).

Observation #4

The horizontal symmetry that occurs, that could allow for a collapse in the perpendicular direction. It could create a smooth ripple like effect if it occurs.

Observation #5

Blocking vs Stitching

Blocking shows to still be both horizontally and vertically in a state where an amplifying combination can occur. However, with stitching it appears to show that the symmetry collapses and a new structure appears that abstractions can be made from.

Observation #6

Plane abstraction

Z - spin/NULL density.

Y - in stitched collapsed state, the idea of parallel differing densities..

X - in stitched collapsed state, the idea of shortened frequency of oscillation along Y due to collapse vs un-collapsed left right oscillation.

Observation #7

Symmetry occurs on almost all planes at all abstractions due to the very nature of working with the least number of instructions:

LEFT

RIGHT

HALT

Observation #8

Consider sigma(0,0) = sigma(+0,+0), sigma(-0,+0), sigma(+0,-0), sigma(-0,-0)