A Di_Alogue Logic

Henry George Skupniewicz

Submitted to the Department of Architecture
in partial fulfillment of the requirements for the
degree of

Bachelors of Science in Architecture

at the

Massachusetts Institute of Technology

June 2013

Signature of Author: _____

Department of Architecture
24 May, 2013

Certified by: _____

George Stiny, PhD
Professor of Design and Computation
Thesis Supervisor & Advisor

Accepted by: _____

J. Meejin Yoon, MAUD
Associate Professor of Architecture
Director of the Undergraduate Architecture Program

A Di_Alogue Logic

Henry George Skupniewicz

Department of Architecture

Submitted to the Department of Architecture on 24 May, 2013, in partial fulfillment of the requirements for the degree of Bachelors of Science in Architecture at the Massachusetts Institute of Technology

Abstract

The history of computation owes a major debt to the traditional crafts, and the worlds of design and computation have been interlinked since the development of mechanical computing systems during the 19th century. As computing systems became digital, the connections between craft and computation have become more abstract, though they are still there.

The regime between the analogue world of craft, and more generally design practice, and the digital world of computation, here referred to as the "di_alogue" world has barely been explored. By challenging our notions of both craft and computation, how can excursions into the di_alogue world help us to re-define or re-conceive of our traditional understanding of craft and of computation? In this thesis, I examine the shared history of traditional craft and computation as well as cover several examples of how these worlds have been combined. Additionally, I argue that by capitalizing on the procedural backbone of a particular craft, one can create unique "logics" that blur the perceived line between craft and computation.

George Stiny, PhD

Professor of Design and Computation
Thesis Supervisor & Advisor

A Di_Alogue Logic

Henry G. Skupniewicz

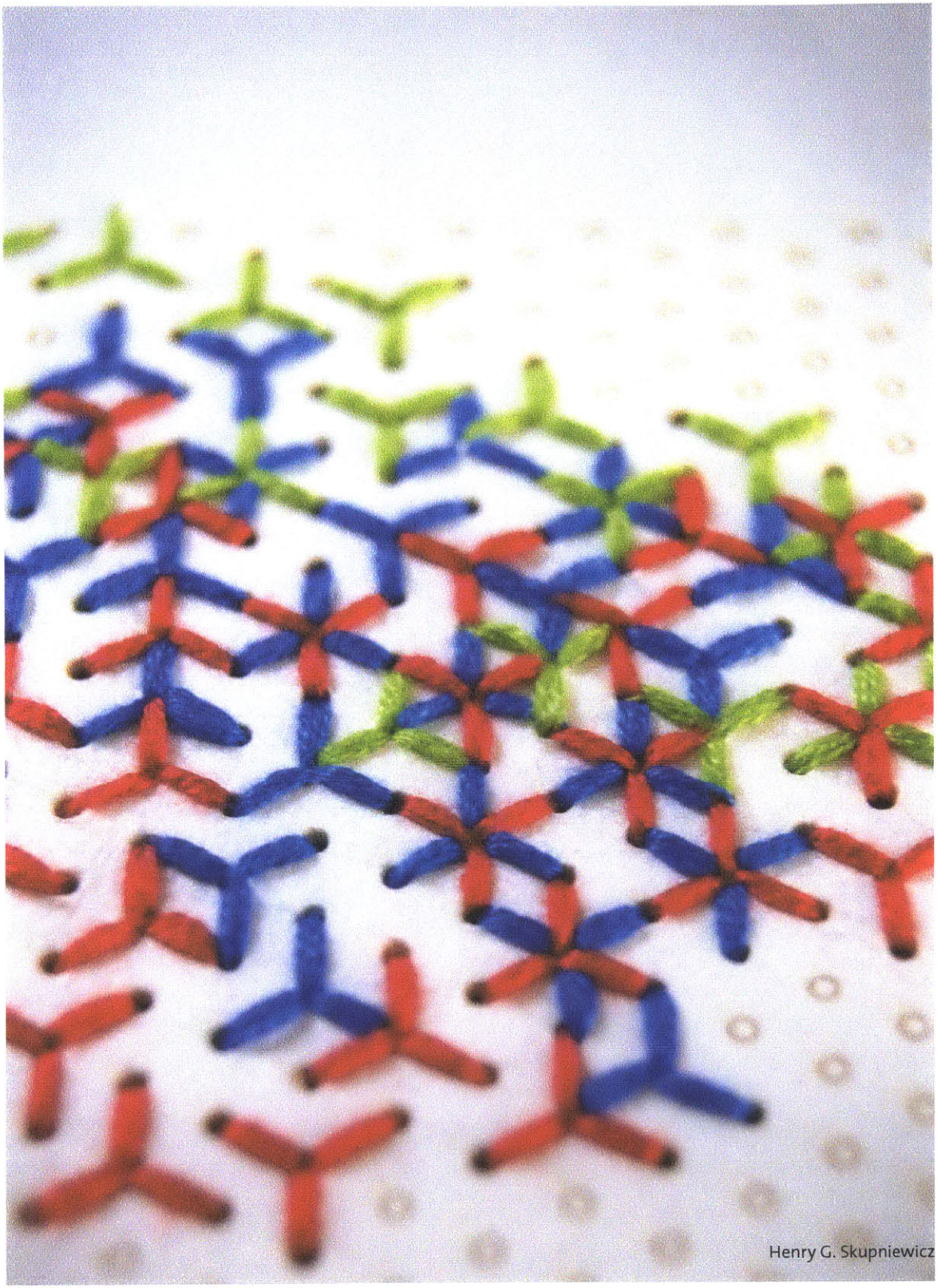

Henry G. Skupniewicz

Table of Contents

3	**Abstract**
11	**Introduction**
17	**Historical Background**
23	**Part I: Printing with Craft**
25	Concepts & Examples
33	Printing with Cross-Stitch
39	**Part II: A Di_Alogue Logic**
41	Concepts, et al.
49	A Di_Alogue Cross-Stitch
63	Bibliography
69	Image Index

Introduction

"'Jacques,' returned Defarge, drawing himself up, 'if madame my wife understood to keep the register in her memory alone, she would not lose a word of it — not a syllable of it. Knitted in her own stitches and her own symbols, it will always be as plain to her as the sun. Confide in Madame Defarge. It would be easier for the weakest poltroon that lives to erase himself from existence than to erase one letter of his name or crimes from the knitted registrar of Madame Defarge.'"

from "A Tale of Two Cities" Charles Dickens

In the following pages I hope to describe a search for a "di_alogue" craft, a medium that transcends both notions of the "digital" or the "analogue," a craft that exists, simultaneously, in both worlds, and in doing so, helps us to re-see and re-examine our conceptions and mis-conceptions of these, seemingly, diametric paradigms.

With the development of such a media, a di_alogue media, I hope to engender discussion both into the roles that traditional craft can play in academia and the possibilities for computation by expanding its purview beyond the world of "ones" and "zeros."

In retrospect, I realize that this quest, the search for a di_alogue craft, has, in some way, been part of my life since I was a small child. While this may seem dramatic, or even silly, I contend that, just as anyone else, my past has profoundly affected my present, and will continue to influence my future. In this way, I can see from the perch I hold now, that this search has appeared, in one way or another, since my youth, no matter how ignorant my position might have been at the time.

Beginnings

Growing up, I always loved puzzles, both of the jig-saw verity, as well as the logic type. Rainy days (and not-so-rainy-days) had me fitting as many puzzles together as I could and, then, lining them up to build a "gallery" of my work. And, when he would come home, my father and I would build massive towers with his childhood set of blocks. Rectangular prisms would become columns or streets sections, that would help to map out the city that we were charged to build. Other times, these blocks would be strewn out over the floor, and then I would meticulously sort and categorize each one — all the squares on this corner of the rug, all the cylinders on that conner.

As I grew older my love of physical puzzles became a love for logical ones. The puzzle boxes that filled my shelves had to make way for games that pitted one player's logical wit against that of another. Tangrams®, Blockus®, RummyCube®, and Othello® took the place of puzzles of garden-scapes and ocean scenes. These games challenged one to fit pieces and tiles together in complex ways, following rules and strategy; each manipulation causing ripples of effects throughout the entire game.

The closet's shelves were not the only ones to become squished for space at this time. My books had to fight for space with the pads and workbooks filled with sudokus and other logic puzzles. Here, symbols were the manipulables, replacing the physical tiles with a representation, with an abstraction. So instead of disks to lay on an Othello board, I took numbers and fit them into a grid upon a page.

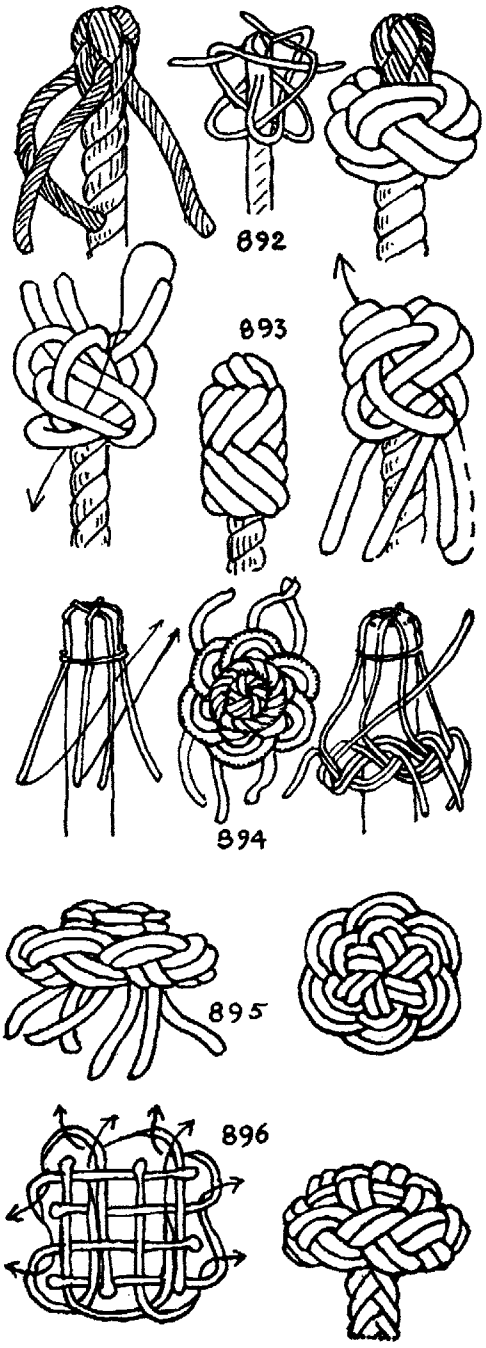


Figure 0: Decorative knotwork from Clifford W. Ashley's *The Ashley Book of Knots*, originally published in 1944.†

I suppose this penchant for solving puzzles and putting things together naturally coincided and influenced my interest in craft, an interest that still makes up a substantial part of my personality today. Although they may not initially lend themselves to this conclusion, crafts have much to do with puzzles and building, and more generally, with the realm of combinatorics. Traditional crafts, such as knitting, embroidery, weaving, basketry, et cetera, are all puzzles or building sets of some sort.

Each craft uses simple elements, combined by simple means, to give birth to brilliantly complex products. For instance, knot-work — really a form of plaiting, or weaving — boils down to rope or fiber, very simple materials. Knot-work transcends the realm of simple shoe-lace-tying and becomes a craft through the purposeful manipulation of said rope. In this way, simple twine can transform into beautiful braids or intricate Gordian knots.

And so, it was this process of taking raw material, the elements, and combining them into a finished product that attracted me to many of the various crafts[1] that I have learned over the years. As a very small child, I took classes in weaving through the local, public-recreation department. In high-school I concerned myself with metal-work. I connected thousands of metal links while making chain maille, and I ground down the edges of glass shards to make them tessellate in the stained glass that I would soldered together. Later, I learned how to spin my own yarn form wool, silk, and other fibers. Yet, the craft that has had the most profound effect on my life, until now, has been that of knitting.

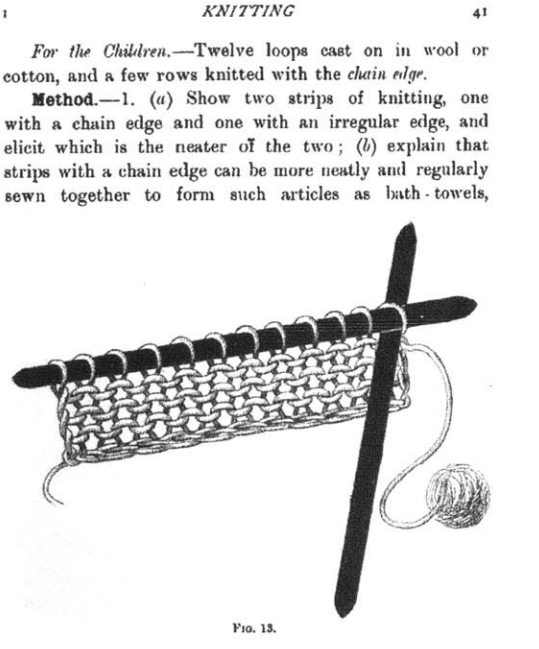
While in eighth grade, in a search for an activity to do during recess, I began knitting. From that point, knitting became more than just a time-filler during recess, it became a passion, a passion that I still have today. Of course I enjoyed the calming aspects of knitting, so too, I enjoyed the social and, even material, outcomes of it, but what I found that really held my interest was uncovering its "code."

Knitting has a fairly long history and, throughout the several hundred years that it has been part of the Western craft tradition, many techniques have developed around the simple premises at the core of the craft: the looping of yarn. Suffice it to say, knitting becomes a chaining together of the fundamental stitches — "knit" stitches and "purl" stitches — along with many of the techniques that have been developed. Through this process of combination, knitters are able to produce a myriad of forms with endless variations of each.

So, while researching the developmental history of the craft, I enjoyed experimenting with new patterns of stitches and stitch-manipulations. I was never satisfied simply following directions or "a pattern" like many of my fellow knitters; I had to figure out why socks were constructed in one way, mittens, another, and hats, a third. I have, at times, spent more times knitting tests and drawing diagrams of stitches then I have actually knitting the sock, or whatever object, I wished to complete.

1 Here I refer to "crafts" in a more general sense.

Years later, solidly settled in at MIT, I was part of a discussion seminar on shape grammars. One day, Professor George Stiny (MIT 67') stated, off-handedly, and I paraphrase, "The choice of ones and zeros, for computing, was completely arbitrary." At the time, this remark gave me pause. Was this one of Professor Stiny's toying generalizations meant to churn up discussion? Or, was it the truth? Regardless of its validity — which is, incidentally, true — I realized that this little statement tapped into some of my long-held inclinations toward puzzles, combinatorics, and, fundamentally, the process of disassembly and re-assembly in the pursuit of knowledge. This question lead to its natural decedents: what if "ones and zeros" are not the best way? What other ways can (or could) we compute, solve problems, and construct? And, even if we are "stuck" computing with our "ones and zeros" how might the process of challenging this paradigm help us re-see our current state?



1 KNITTING 41

For the Children.—Twelve loops cast on in wool or cotton, and a few rows knitted with the *chain edge*.

Method.—1. (*a*) Show two strips of knitting, one with a chain edge and one with an irregular edge, and elicit which is the neater of the two; (*b*) explain that strips with a chain edge can be more neatly and regularly sewn together to form such articles as bath-towels,

FIG. 13.

dusters, caps, cuffs, bed-quilts, cushion-covers, kettle-holders, etc.

2. (*a*) Write *chain edge* on the slate; and (*b*) tell children this edge is so called, because it looks like the links of a chain.

3. Draw an illustration on the slate of the piece of knitting given to the children (Fig. 13).

Figure 1: A page from *A Text-Book of Needlework, Knitting, and Cutting-Out*, published 1893.†

In my attempt to answer or, better, address these questions, I have begun to call back onto my historical love and knowledge of craft by re-examining their foundations as well as going back to the fundamentals of our current computational methods.

I have realized that many traditional crafts can be described as computational enterprises; crafts such as knitting, crochet, embroidery, and weaving all instruct practitioners to combine many discrete "units" (whether physically discrete or conceptually discrete) to combine and form, as a whole, a separate object. For example, rows and columns of individual stitches form knitted fabric, each of which is individually manipulated by the knitter. Weaving has several layers of this "atomization" effect. First, and most superficially, woven cloth is the aggregation of inter-twined threads. But, at a deeper level, each of these threads have a specific path over and under the others which may be considered as a series of discrete operations which navigate each thread through its route.

This atomization of craft is very similar to the way that we, as humans, have chosen to compute, and more importantly, compute digitally. Our seem-

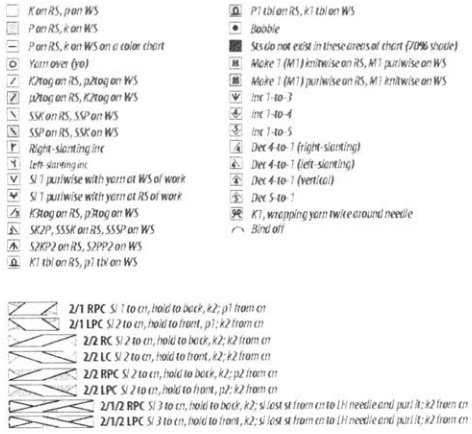


Figure 2: Symbols used in charted knitting designs.†

ingly complex digital universe is only amalgamation of two values ('one' and 'zero') and a handful of operations which can be applied to these values (AND, OR, NOT, XOR, NAND, et cetera.).

While is may seem inconceivable that the complexity that we see in the architecture, design, manufacturing, and entertainment industries could derive from anything this simple, we only have to look at the pinnacles of examples of craft, 'analogue' or physical processes, to locate similar displays of bewildering intricacy within knitting or embroidery; all of which were made through a series of small, simple gestures.

No.	Truth Table	In Words	PWR	Polish	McCulloch	Logic Alphabet
1	FFFF	Contradiction	Contradiction	O	X	o
2	FFFT	Not-A and Not-B	~ A · ~ B	X	X	p
3	FFTF	Not-A and B	~ A · B	M	X	b
4	FTFF	A and Not-B	A · ~ B	L	X	q
5	TFFF	A and B	A · B	K	X	d
6	TTFF	A	A	I	X	c
7	TFTF	B	B	H	X	u
8	TFFT	A equivalent B	A ≡ B	E	X	s
9	FTTF	A or else B	A ⋀ B	J	X	z
10	FTFT	Not-B	~ B	G	X	n
11	FFTT	Not-A	~ A	F	X	ɔ
12	FTTT	Not-A or Not-B	A/B	D	X	h
13	TFTT	if A, then B	A ⊃ B	C	X	ɥ
14	TTFT	if B, then A	B ⊃ A	B	X	ɾ
15	TTTF	A or B	A ⋁ B	A	X	ɥ
16	TTTT	Tautology	Tautology	V	X	x

Figure 3: Traditional logic symbols use in formal logic.†

In the following pages I will track the conjoined development of out current computation system with that of traditional craft and explore and review ways in which people have either exploited, explored, or hacked the world between those of computation (digital) and of craft (analogue).

This hacking, when done in a particular way in which one side does not dominate and which plays to the strengths of each system, breaches the world of the Di_Alogue.

In addition to this, I will also examine several unconventional computational techniques that achoo the conventions that run our computers and other computing systems now. By looking at examples from the world of computer science that question and re-examine the fundamentals of the theory of computation, I extract some themes that offer solutions, or at the very lest, hint at solutions, to the problem of combining craft (and design) and computation in a "seamless" way.

With all this information, I will propose an example system of logic which exists in conjunction with the world of cross-stitch, a traditional craft technique. By utilizing the logic and rules that already exist within the medium,[1] I am able to demonstrate many traditional logical functions within a semi-traditional execution of cross-stitch, thus combining the digital and analogue (or computation and craft) worlds in a novel way yet to be demonstrated.

1 Here "medium" refers to both the constraints and considerations afforded by the physical limitations and abilities of the system as well as those that come from tradition and convention within the craft.

Historical Background

The histories of craft and digital computation are, surprisingly, intertwined. What is more, the later is, ultimately, in debt to the former.

This shared history finds its origins in revolutionary France where a booming silk industry untilized highly complicated looms, called *draw-looms*, that were capable of weaving complicated patterns and designs within the luscious fabric by allowing one of the operators to specify unique arrangements of the threads.[1] Yet, as Essinger states in his book, *Jacquard's Web*,

> "The real problem with the draw-loom was that is was not a *machine* at all. Instead, it was really only a device for facilitating the *manual* weaving of patterns or images into fabric."[2]

In order to address this problem, which resulted in slow production rates, several inventors attempted to automate, or semi-automate, the process. The first attempts at mechanical automation happened in the 1720s. Basile Bouchon, in 1725, built a loom that configured strings based on holes punched in a roll of paper (see Figure 4).[3] In 1728, an inventor with a last name of Falcon, experimented with looms semi-automated by punched cards; these experiments did not bare much fruit.[4]

The first inventor to make a significant contribution to the automation of the draw-loom was

1 Essinger, James. *Jacquard's Web: How a hand loom led to the birth of the information age.* Oxford: Oxford University Press, 2004. Print. 17.

2 ibid.

3 Held, Shirley E.. *Weaving: A Handbook of the Fiber Arts.* 2nd ed.. New York: Holt, Rinehart and Winston, 1978. Print. 97.

4 Essinger, 36.

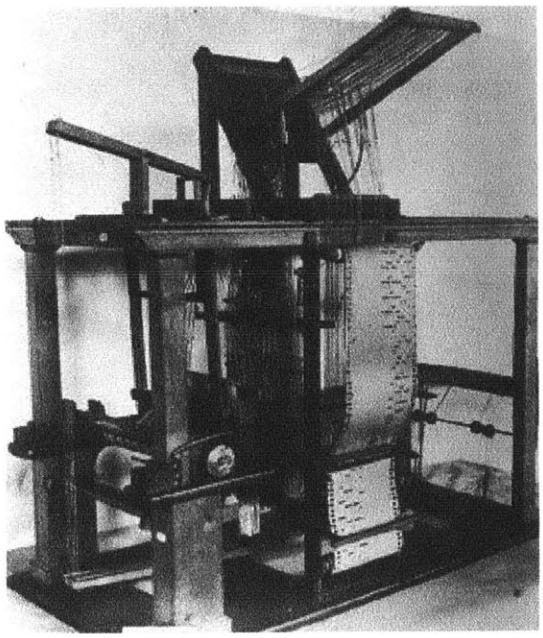


Figure 4: Bouchon's paper-tape guided loom.†

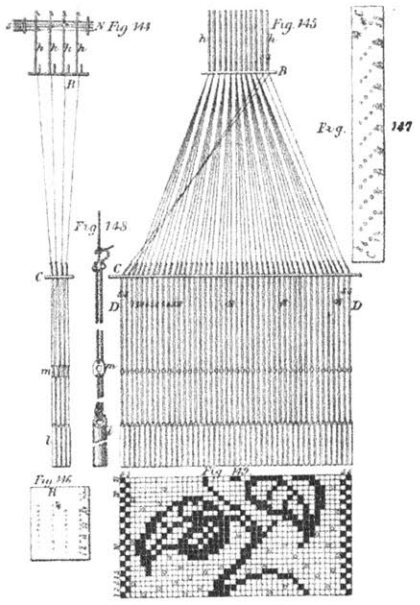


Figure 5: Jacquard's loom.†

Jacques de Vaucanson. De Vaucanson, in 1745, invented a loom that replaced Falcon's punched cards with a metal drum, similar to those in a music box.[1] Essinger explains, though, that

> "...the metal cylinders were expensive and difficult to make. Moreover, by their very nature they could only be used for making images that involved regularly repeated designs."[2]

It was not until sixty years passed that someone addressed these problems; that person was Joseph Marie Jacquard.

In 1804, Jacquard finished designing a loom that was commissioned by the French government to address the defects in the de Vaucanson design.[3] Jacquard's loom went back to Falcon's idea of using punched cards; but where Falcon failed, Jacquard excelled.

Jacquard's loom (see Figures 5 & 6), which has appropriated the apt name of "the Jacquard loom," was able to fully automate the process of manipulating threads on a loom to allow for any image to be woven. Additionally, in the way he set up the interface between the loom and the punched cards, the cards automatically loaded themselves into the contraption.[4] At the time of its invention, "it was unquestionably the most complex mechanism in the world."[5]

About thirty years after its invention, the Jacquard loom departed from its singular life as a tool of craft and became an inspiration for the entire digital, computational age. Charles Babbage, who is considered by many as the grandfather of the digital age, invented, at least theoretically,

1 ibid., 17-18.
2 ibid.
3 Essinger, 37. Held, 97.
4 Essinger, 37.
5 ibid..

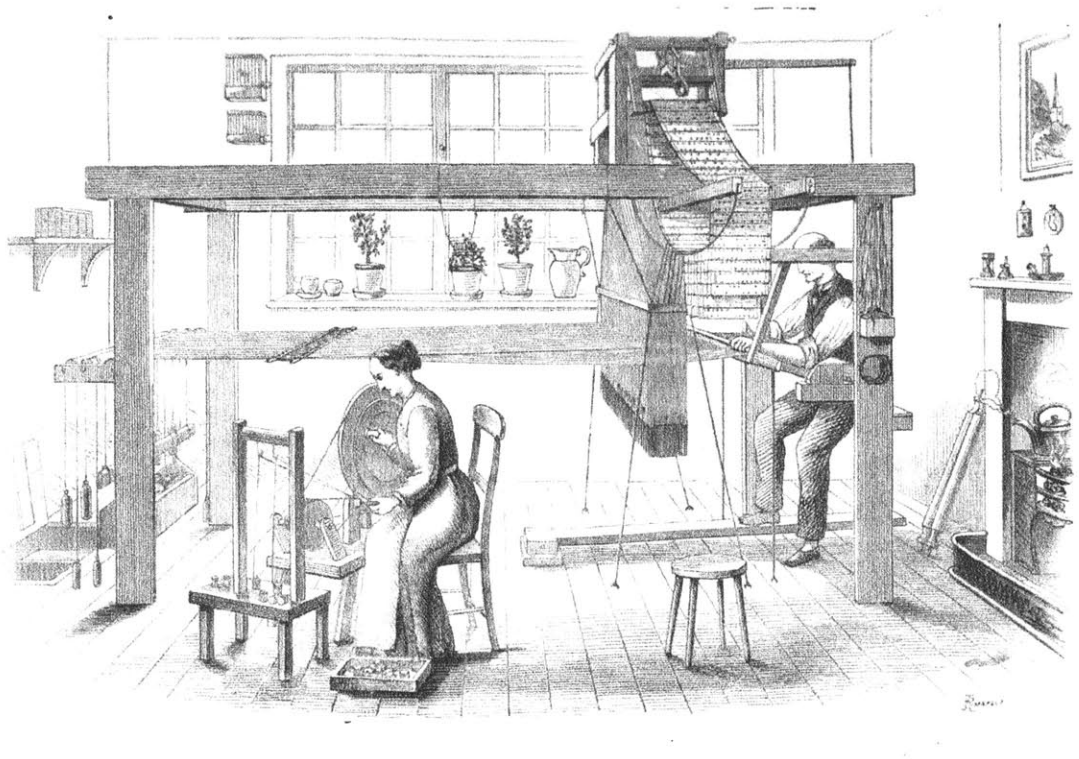


Figure 6: A Jacquard loom in use.†

what has been considered *"a massive Victorian computer, made out of cogwheels."*[1] While it is wildly known that Babbage's Difference Engine (See Figure 7) and Analytical Engine (see Figure 8), or, at the very least, the theory behind and designs of them, greatly, and directly influenced future advances in computing, what is less known, and important for our purposes here, is the direct connection between Babbage and Jacquard.

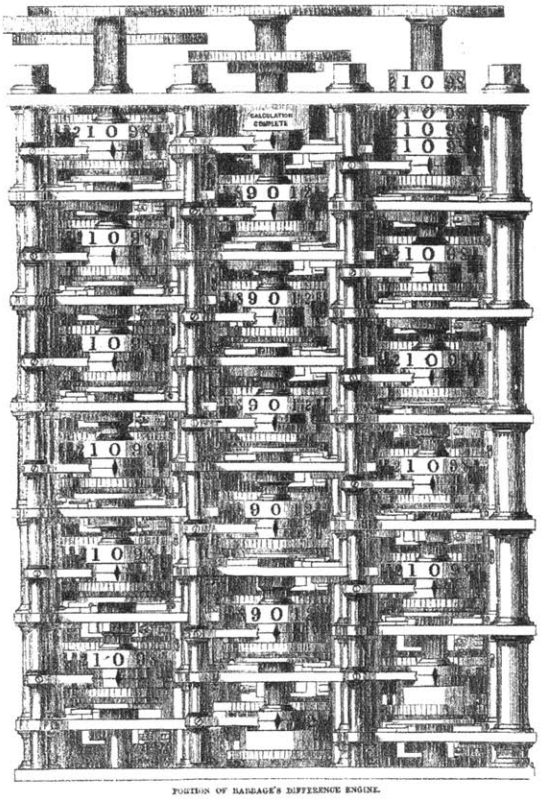
In 1836, when designing his Analytical Engine, a programmable, calculating machine, Babbage notes in his notebooks "'Suggested Jacquard's loom as a substitute for the drums.'" in his discussion of input devices.[2] Babbage would later, in 1864, make particular reference to Jacquard and the weaving process in his autobiography, *Passages from the the Life of a Philosopher.*[3] Thus, we can see a direct line beetween the traditional craft of weaving to what many concider to be the first "computer," with that word used in a very similar way to our contemporary notions of the word.

After Babbage, the progression to our current sate of computing rapidly progresses. The histories of IBM, code-breaking, and the theoretical founders of the computing age are widely reasearched and writen about. It is not the purview of this thesis to explore those areas, but I would heavily recommend reading both the history as

1 ibid., 84.

2 ibid., 85.

3 ibid..



PORTION OF BABBAGE'S DIFFERENCE ENGINE.

Figure 7: Babbage's Difference Engine.†

well as the seminal papers that defined the computing era. Much of the theoretical structure of computing, such as cellular automata, though it can focus of abstract concepts, also touch upon visual creations and would be useful to designers. Many of the works in the Bibliography at the end of this work, fit into this class of reading and would be very useful to those interested.

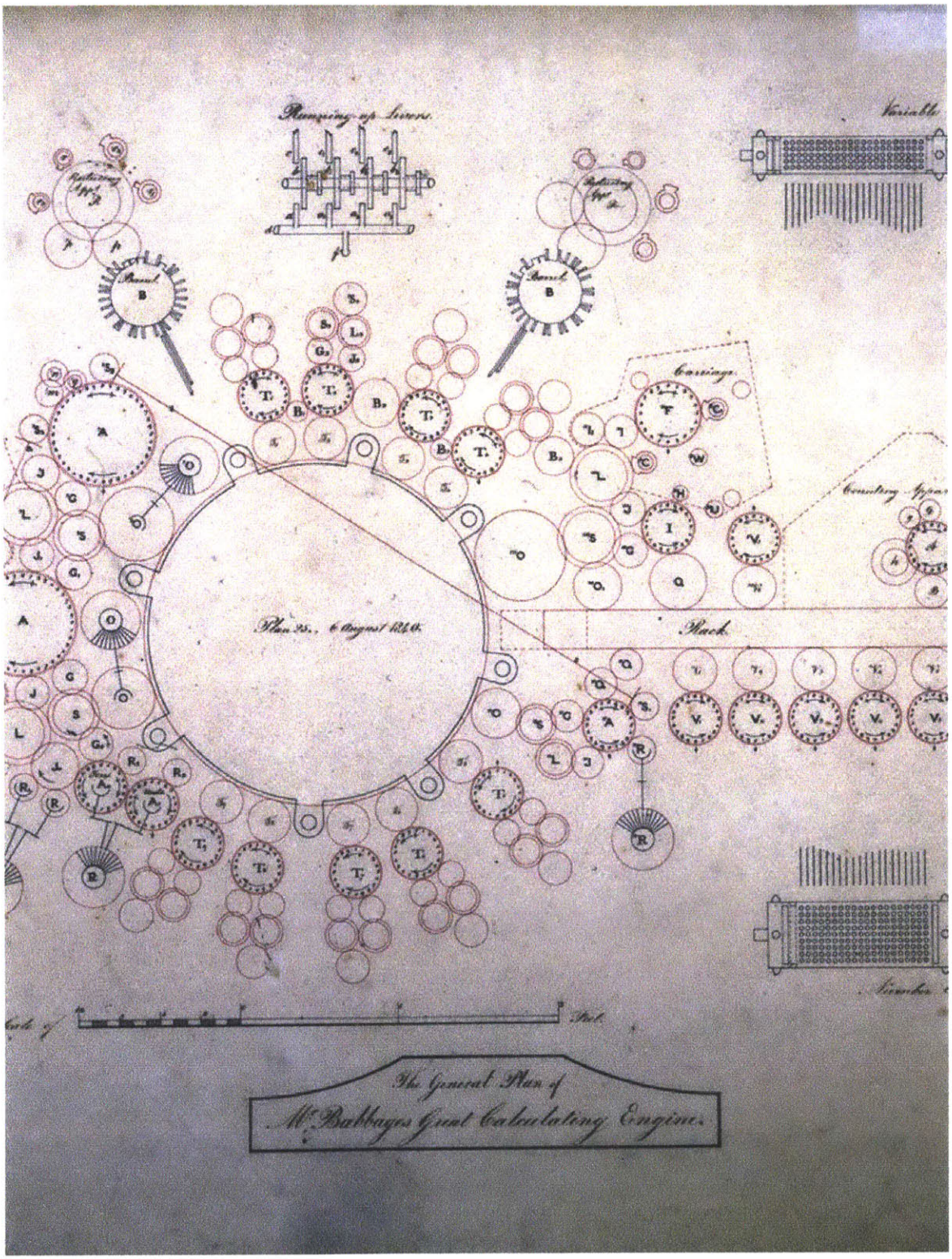


Figure 8: Babbage's plan for the Analytical Engine.†

Part I
Printing With Craft

Concept and Examples

Up until this point the majority of research done into the shared world of craft and computation has taken a form that I describe as "printing with craft." In the following section I will define this concept as well as explore several examples of work that "print with craft," while looking to/into the Di_Alogue world.

When someone "prints with craft", they project digital concepts or ideas onto and with the medium of a craft; the craft's own logic have no real or significant effect on the functionality of the object other than to provide novelty. Because of this work derived in this way is not truly "di_alogue" in the strict sense — a convoluted interplay between the digital and the analogue is missing — though it would be unfair to state that projects of this sort cannot *suggest* or *posses* certain characteristics of true di_alogue systems. In this way, looking at them, and understanding how they work physically and conceptually as well as how they interface with craft and computation can offer insights into the barely seen world of the di_alogue.

Work in the realm of "printing with craft" is not overly scarce, in fact, there seems to be quite a bit of interest in it. I would suggest that this interest is derived from the conspicuous nature in which computation is projected through a craft medium. This shifting perspective makes both of these common systems apparent, which is in contrast with their ubiquitous nature. LEDs are everywhere, so, too, are knit sweaters; but hold onto your hats if LEDs are knitted into your new pullover!

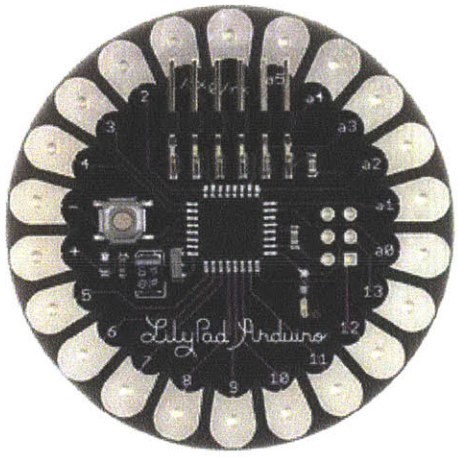


Figure 9: The LilyPad Arduino.†

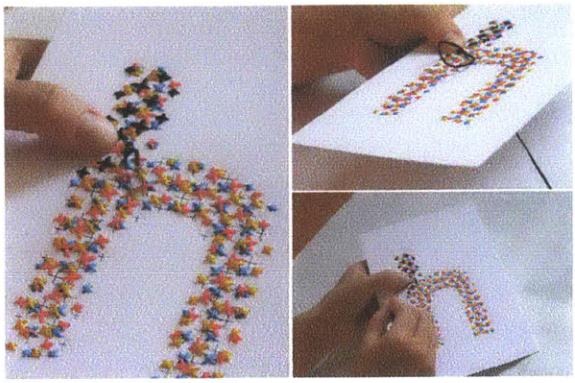


Figure 10: Kasikov in production.†

LilyPad Arduino

One of the most famous examples of this mode is found within the exploration of "soft circuits," or textiles the incorporate electronic components. Leah Beuchley of the MIT Media Lab's High Low Tech Lab has become an expert on the incorporation of electronic components into fabrics, paper, and other "crafting" media. As part of her research she has developed the LilyPad platform (see Figure 9), a version of the popular Arduino that is designed to interface with conductive threads rather than the traditional wire.

Yet, like the tongue-in-cheek example above, the LilyPad, and other examples of Beuchley's work, only "project" or "print" circuits onto, or through, the textiles and other media that she uses. And, in these cases, the media, the craft, can only offer a novel application or interpretation of traditional electronics. The nature of the construction of a running stitch that might hold a shirt together but also act as a conductive wire, really completes its two tasks separately, there is little discussion between the two systems (the analogue stitching and the digital connectivity).

CMKY Embroidery

Evielin Kasikov presents another novel incorporation of craft with the digital technology of offset printing. Kasikov, a graphic designer, "prints" her designs with embroidery floss rather than ink , in a process she has dubbed "CMKY Embroidery"(-See Figures 10 & 11). Her designs are quite captivating as it makes both the printing technology as well as the embroidery conspicuous, thus prompting discussion.

Unfortunately, this discussion really only reframes current notions by offering new ways to make physical our digital data, and in a way that, in essence, only copies current print paradigms. Additionally, in order to do this, she must set aside much of the tradition behind counted cross-stitch. As can be seen in Figure 11, none of the stitches comply with the standard grid typically associated with cross stitch. The form of the embroidery floss upon the paper that she uses (See Figure 10), is enough to make the connection for her audience, after this point she does what

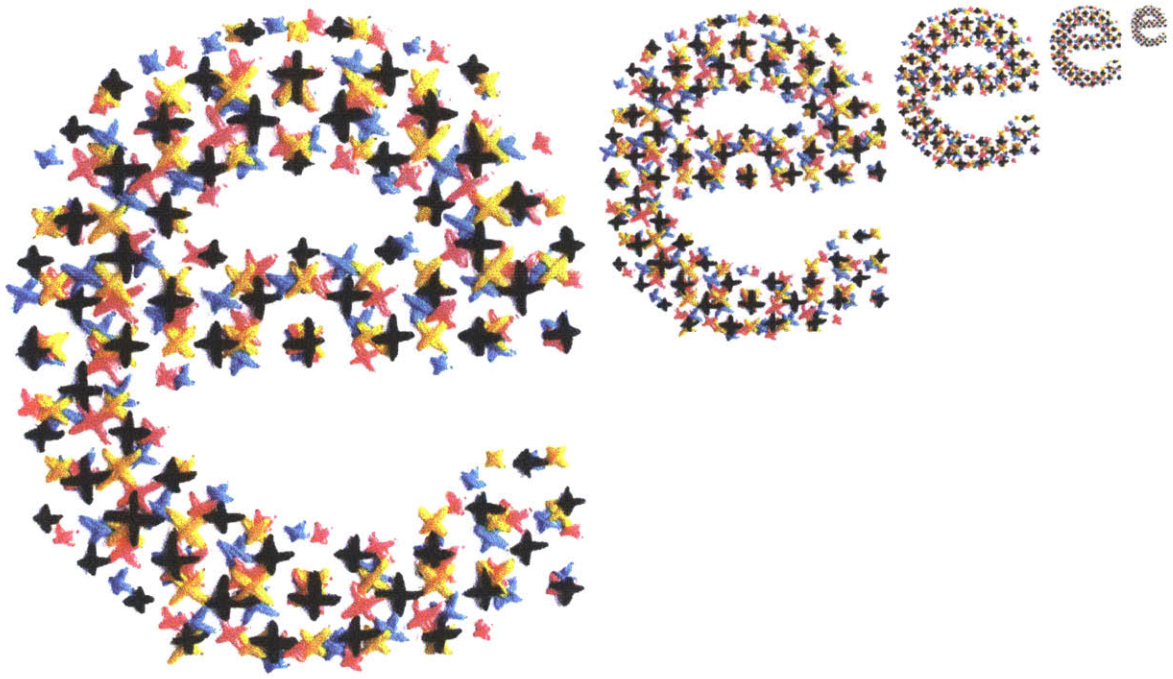


Figure 11: An "e" in CMKY embroidery.†

she wishes with the rule "intrinsic"[1] to cross-stitch and then focuses on the "digital" aspects of her piece.

Computational Justification in Weaving

There are several examples of people using computation and mathematics in conjunction with the craft of weaving. They have, as a whole, used these areas of study, in what can be best described, as compositional, or design, justification.

Jer Thorp, an Adjunct Professor at New Your University, has created a website with a Processing script which runs continuously that "weaves" an infinitely long piece of fabric (see Figure 12). His "Infinite Weft" takes directions, pattern-wise, from rules based on Stephen Wolfram's cellular automata.

Another notable character: Ada Deitz, a weaver in the mid-20th century, used algebraic expressions to guide her decisions of how to set up her loom and how to proceed with the weaving.[2]

Again, in a very similar way to that of the CMKY Embroidery, nether Thorp nor Deitz directly address the logic of weaving itself: the fact that weaving is a physical, constructive system with both physical , as well as historical, "rules" and constraints. This is not to even mention fabricatablity.

Finally, Ralph E. Griswold, a former professor of computer science, followed a similar path to Ms. Deitz. After his retirement, he immersed himself

1 In the sense that tradition, history, and convention can embed something with intrinsic features.

2 Dietz, Ada K.. *Algebraic Expressions in Handwoven Textiles*. Ed. Robert Kirkpatrick. Louisville, KY: Little Loomhouse, 1949. Web PDF. 23 Nov, 2012.

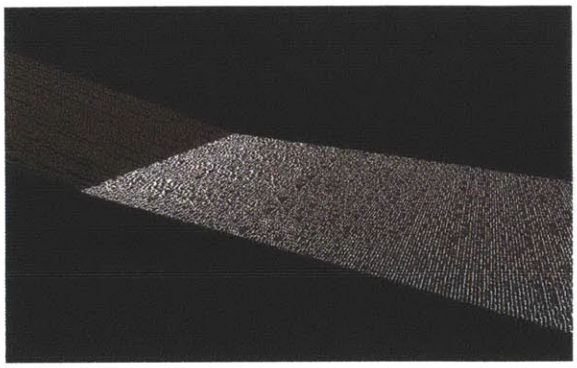


Figure 12: "Infinite Weft."†

in the world of weaving and chose to look at it through his own eyes, which were trained as a computer scientist. In his collected works, *Mathematical and Computational Topics in Weaving*, he analyses weaving patterns mathematically and describes ways to produce them using computational inspiration.

> It is telling, though that, in the preface to the book, he states, "I realize that by not being an actual weaver, I am missing a great deal and that there are gaps in my knowledge and understanding."[1]

So, in actuality, most of Griswold's analysis deals with the graphical and notional aspects of weaving, which, to his own admission, leaves much to be desired. This is because weaving, like every other craft, is so process-based, that leaving out the act of producing with the craft throws away a majority of the essence of the media.

As a computer scientist/mathematician, Griswold is a perfect segue to the next subsection which address those who have turned "printing with craft" on its head: those who have ad-

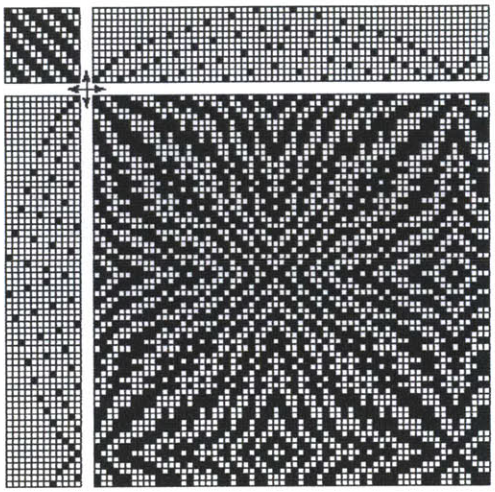


Figure 13: One of Griswold's designs in standard weaving notation.†

dressed mathematical questions with the help and inspiration of craft, rather than craft being inspired by computation.

Computation Inspired By Craft

There is a small collection of works that address mathematical and computer science question using craft procedures and work as an inspiration or as an analogy. A fairly comprehensive listing of the non-weaving related published, scholarly articles can be found within *Making Mathematics with Needlework*.[2]

Other articles addressing craft can be found within the purview of computational geometry, which can look at threads and embroidery floss, as physical realizations of abstract graph-theory

1 Griswold, Prof. Ralph E. *Mathematical and Computational Topics in Weaving*. Self-published, 2006. Web PDF. 23 Nov, 2012. xii.

2 Belcastro, Sarah-Marie & Carolyn Yackel, eds.. *Making Mathematics with needlework: Ten Papers and Ten Projects*. Natick, MA: A K Peters, Ltd., 2008. Print.

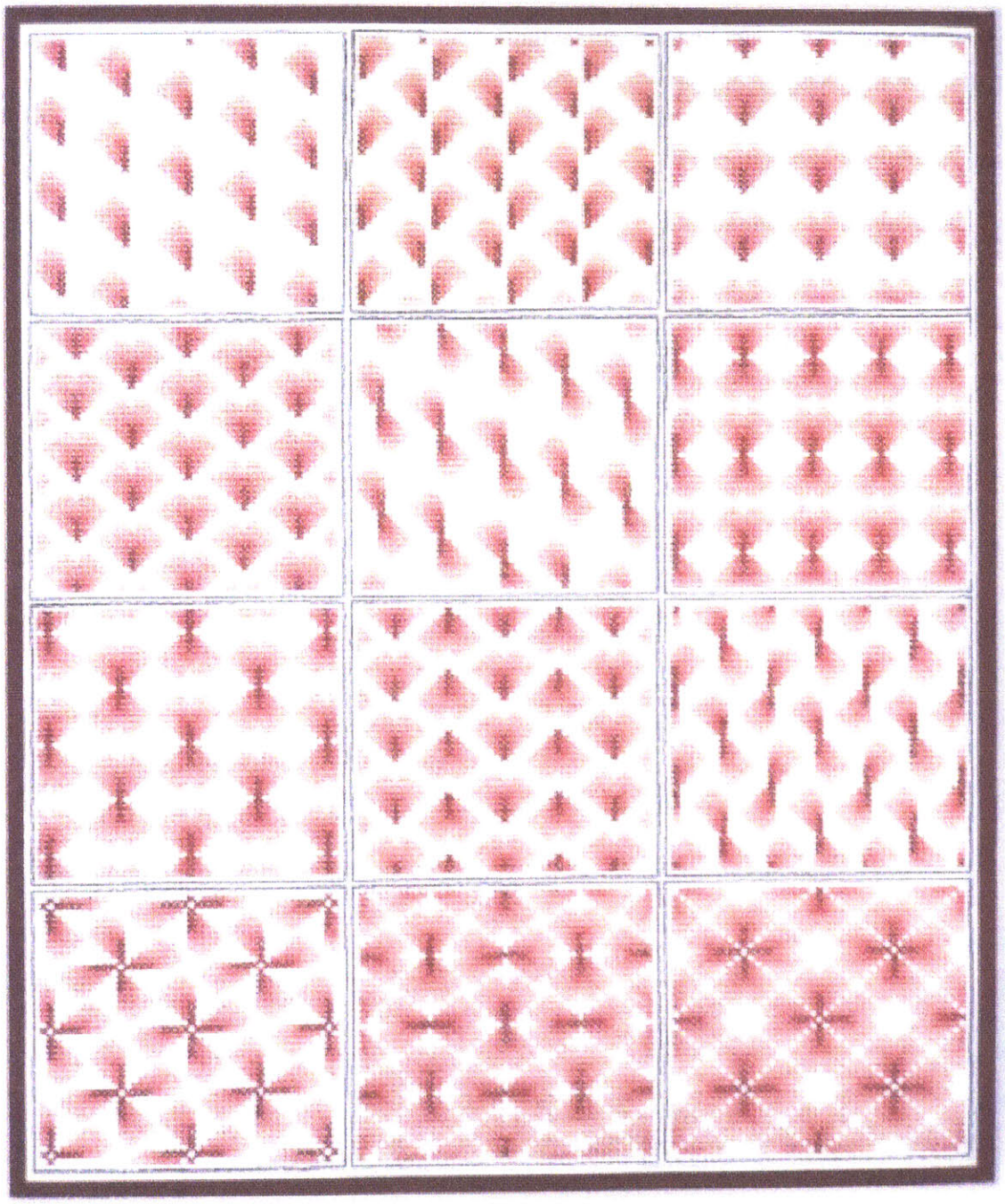


Figure 14: Pattern sampler.†

problems.[1, 2]

For example, in *Making Mathematics with Needlework*[3] one of the contributers, Mary D. Shepherd, looks to displaying the various types of symmetric within a cross-stitch sampler (see Figure 14) in her article "Symmetry Patterns in Cross-Stitch." This paper *does* review the difference between "theoretical" patterns for a cross-stitch design that, on paper and conceptually are different, and "real" produced designs which might, when looking at the finished products superficially, simplify the set of possible answers to some design space. The address of this aspect starts to bring this project more in line with true di_alogue concepts.

1 Arkin, Esther M., et al. "The Embroidery Problem." *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG2008).* Montéal, Québec, August 135-138, 2008. Print

2 Biedl, Therese, et al.. "Cross-Stitching Using Little Thread." *Proceedings of the 17th Canadian Conference on Computational Geometry (CCCG'05).* 199-202. (2005). Print.

3 Belcastro.

Printing With Cross-Stitch

With the inspiration of the owrk already seen, as well as a drive to see a project that more-fully integrates the digital world of computation with the analogue world of traditional craft, I have completed several exercices with cross-stich to explore some of the areas discussed.

Cross-stitch is a type of embroidery which "consists of two slanting stitches, one over the other and crossing in the center."[1] These two slanting stitches which are connected via thread running along the back of the fabric, usually in horizontal or verticle dirtections (i.e. not "slanting) to form one entire stitch. These entire stitches are linked with many more using non-slanting stitches running along the back of the fabric, as well. Half-cross-stitch is very similar, but instead of two slanting stitches, it utalizes only one to for one entire stitch.

Cross-stitch also has many "digital" aspects. First, and foremost, it employs a small number of "allowable" stitches. Second, hundreds, if not thousands, of these stitches aggregate to form complex, global patterns. Also, the visual output of hal-cross-stich, where individual stitches either move from a "higher" point to a "lower" point, or vise versa, make it very translatable to an interpretation of traditional, binary output.

In the following section, I will outline the ecercses that I have undertaken to explore new ways to "print with cross-stich," or project digital/computational concepts through this craft.

1 de Dillmont, Th.. *The complete Encyclopedia of Needlework.* Philadelphia: Running Press, 1974. Print. 88.

```
>>> input_string= 'di_alogue'
>>> F= Fabric(logic_mode= 'binary_write',
    input= input_string)
>>> F.weaveFabric()
>>>
>>> \//\\/\\
>>> //\\/\\\
>>> /\\/\\\/
>>> \\/\\\//
>>> \/\\\//\
>>> /\\\//\/
>>> \\\//\/\
>>> \\//\/\\
>>> \//\/\\/
```

Code 0: Fabric() translating "di_alogue" to a cross-stitch representation of the 8-bit encoding of the text.

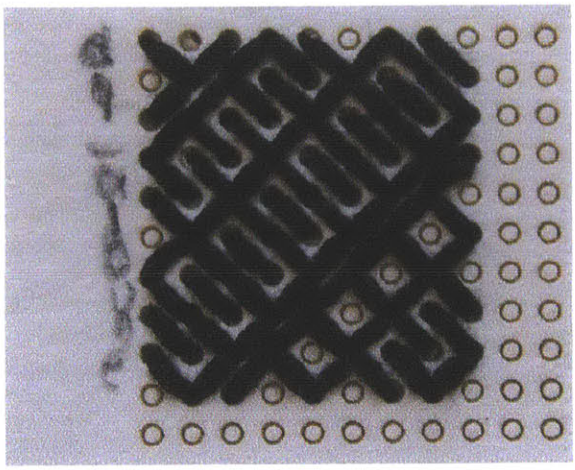


Figure 15: The output of Fabric() from [Code 1] stitched into card-stock with cotton embroidery floss.†

Fabric()

I have written a small Python script I have called `Fabric()` which is able to take several types of inputs, and output several representations, depending on a chosen "mode," of those inputs in the context of half-cross-stitch.

One of my favorite modes of operation involves inputting a text phrase, such as "di_alogue," and receiving, form the code a series of lines of characters that map to the bit-representation of that text. The character representation that it outputs takes the form of a slashes that represent types of stitches which map to the bit data ("0" or "1"). A forward-slash (/) maps to a "1" and represents an embroidered stitch that goes from the lower, left-hand corner of a cell (within the greater cross-stitch grid) to the upper, right-hand corner. A back-slash (\) maps to a "0" and represents an embroidered stitch that goes from the upper, left-hand corner of a cell, to the lower, right-hand corner.

I have then actually embroidered several of these "patterns" onto card-stock (see Figure 15). It is interesting to compare the computer output (See Code 0) and the embroidered "output" as they are identical representationally, but very different visually. Additionally, both are very different form the 1/0 representation.[1] And, of course, the test string can be much longer than 9 characters (see Code 1). (For the full code of `Fabric()`, see Code 2.)

Showing logic

I have also experimented with representing simple binary logic in the language of embroidery (see Figure 16). In these exercises, I have pairs of stitches (two consecutive stitches) representing

1 For comparison, the 1/0 output of the word "di_alogue" is:

```
>>> 01100100
>>> 11001000
>>> 10010001
>>> 00100011
>>> 01000110
>>> 10001101
>>> 00011010
>>> 00110100
>>> 01101001
```

```
>>> input_string= 'The Massachusetts
    Institute of Technology'
>>> F= Fabric(logic_mode= 'binary_write',
    input= input_string)
>>> F.weaveFabric()
>>>
>>> \/\/\/\\
>>> /\/\/\\\
>>> \/\/\\\/
>>> /\/\\\//
>>> \/\\\//\
>>> /\\\//\/
>>> \\\//\/\
>>> \\//\/\\
>>> \//\/\\\
>>> //\/\\\\
>>> /\/\\\\/
>>> \/\\\\//
>>> /\\\\//\
>>> \\\\//\\
>>> \\\//\\/
>>> \\//\\/\
>>> \//\\/\/
>>> //\\/\/\
>>> /\\/\/\\
>>> \\/\/\\/
>>> \/\/\\/\
>>> /\/\\/\\
>>> \/\\/\\\
>>> /\\/\\\\
>>> \\/\\\\\
>>> \/\\\\\\
>>> /\\\\\\/
>>> \\\\\/\
>>> \\\\/\\
>>> \\\/\\/
>>> \\/\\//\
>>> \/\\//\/
>>> /\\//\/\
>>> \\//\/\/
>>> \//\/\//
>>> //\/\//\
>>> /\/\//\\
>>> \/\//\\\
>>> /\//\\\\
>>> \//\\\\/
```

Code 1: Fabric() translating "The Massachusetts Institute of Technology" to a cross-stitch representation of the 8-bit encoding of the text.

either a "True" or "False" ("1" or "0") value or an "AND" or "OR" operator. The first two lines are the a series (in Figure 16) of "inputs" ((1, 0, 1, 0)(1, 1, 0, 0)) The third line is a string of operators (AND, AND, OR, OR) to act on the two values above it. The final line are the "outputs" of the operations

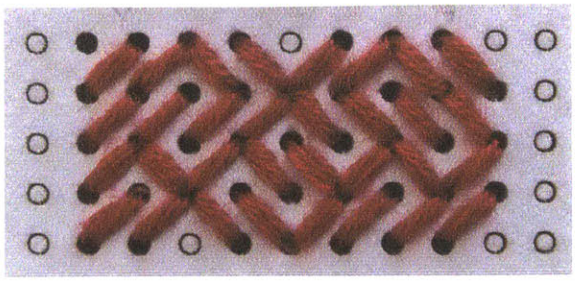


Figure 16: Simple binary logic using AND and OR operators represented using embroidery floss.†

(1, 0, 1, 0 [error in embroidery resulted in a "1"]). This technique of inputing stitches as arguments in a funtion, inputing other stitches are the function itself, and then stiching out the value will be unpacked later on.

Rule 30

Another exercise that I embraked on was looking at the very famous and well-know cellular automata, Rule 30 (see Figure 17). I represented its binary-colored cells in vaerious ways using crpss-stitch as well as half-cross-stitch (see Figuers 18, 19, & 20).

While the interpretation in traditional crss-stich resembles the standard represention o f Rule 30 (see Figure 18), the interpretations of the system into half-cross-stitch start to reveil com intestinf

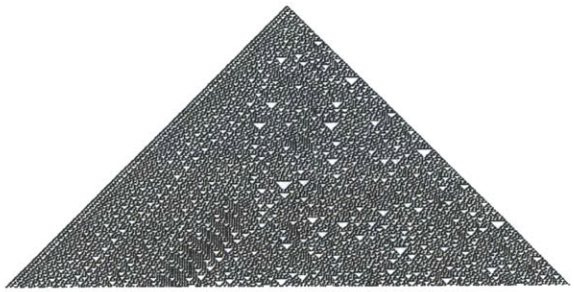


Figure 16: "Rule 30."†

features and have an identity of their own. In Figure 19, sections of Rule 30 are stitched where (clock-wise from left) "upwards stitches" relate to black squares (really "ones" (1)) and the non-stitch to white squares (really "zeros" (0)); "downward stitches now relate to "ones;" and "upward stitches" relate to "ones," "downward stitches," to "zeors."

In all these examples, though, the computational concept of cellular automata is being communicated through the visual aspects of cross-stitch aloone, rather than trhough some interpretation of the under-laying logic of the craft.

```
"""
di_alogue weave: 0.1
Sunday, 1 December, 2012
 |      Henry George Skupniewicz
(||     MIT Dept. of Arch.
 ||)    Design and Comp. Group
  |     hskup@mit.edu  |  henryskup.com
"""
import binascii
 class Fabric(object):
     """
      Stitch Modes ('stitch_mode'):
      'cross-stitch': uses Xs (X) (mapped to 1/True) and blank-
    spaces ( )(mapped to 0/False)
      'diagonal-binary': uses forward slash (/) (mapped to 1/
    True) and backslash (\)(mapped to 0/False)
      'diagonal-trinary': uses forward slash (/) (mapped to 1/
    True), backslash (\)(mapped to -1/False), and blank-spaces (
    ) (mapped to 0/Unknown)
      """
      def __init__(self, row_length= 10, fabric_length= 10,
    stitch_mode= 'diagonal-binary', logic_mode= 'binary_write',
    input= None):
          self.n= row_length
          self.m= fabric_length
          self.mode= [stitch_mode, logic_mode, input]
          self.pattern= None
          self.setFabric()
      def setFabric(self):
          input_string= self.mode[2]
          self.pattern= self.formatToBinary(input_string)
      def formatToBinary(self, input_string):
          binary_string= bin(int(binascii.hexlify(input_string),
    16))
          binary_string= binary_string.replace('b', '')
          return binary_string
      def weaveFabric(self, file_name= 'cross.txt'):
          """Goes through data and displays weave in file."""
          f= open(file_name, 'w')
          if self.mode[1] == 'binary_write':
              stitches= self.weaveBinary()
          elif self.mode[1] == 'cross_automata':
              stitches= self.weaveCA()
          f.flush()
          f.write(str(stitches))
          f.close()
      def weaveBinary(self):
          stitches= ''
          for i in range(len(self.pattern)/8):
              row= self.pattern[i: (i + 8)]
              row_string= ''
              for stitch in row:
                  if int(stitch) == 1:
                      row_string += '/'
                  else:
                      row_string += '\\'
              stitches += row_string + '\n'
          return stitches
      def weaveCA(self):
          return None
  class Row(object):
      def __init__(self):
          pass
  class Stitch(object):
      def __init__(self, int_state= True):
          self.state= int_state
      def getState(self):
          return self.state
```

Code 2: Fabric().

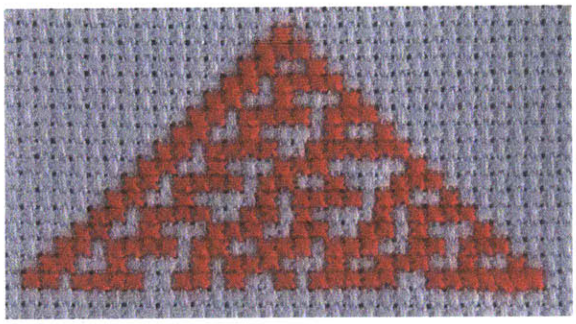


Figure 18: "Rule 30" in traditional cross-stitch.†

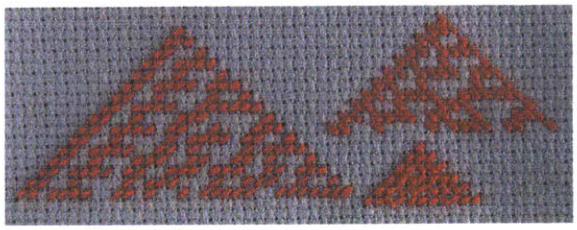


Figure 19: Various interpretations of "rule 30" in half-cross-stitch.†

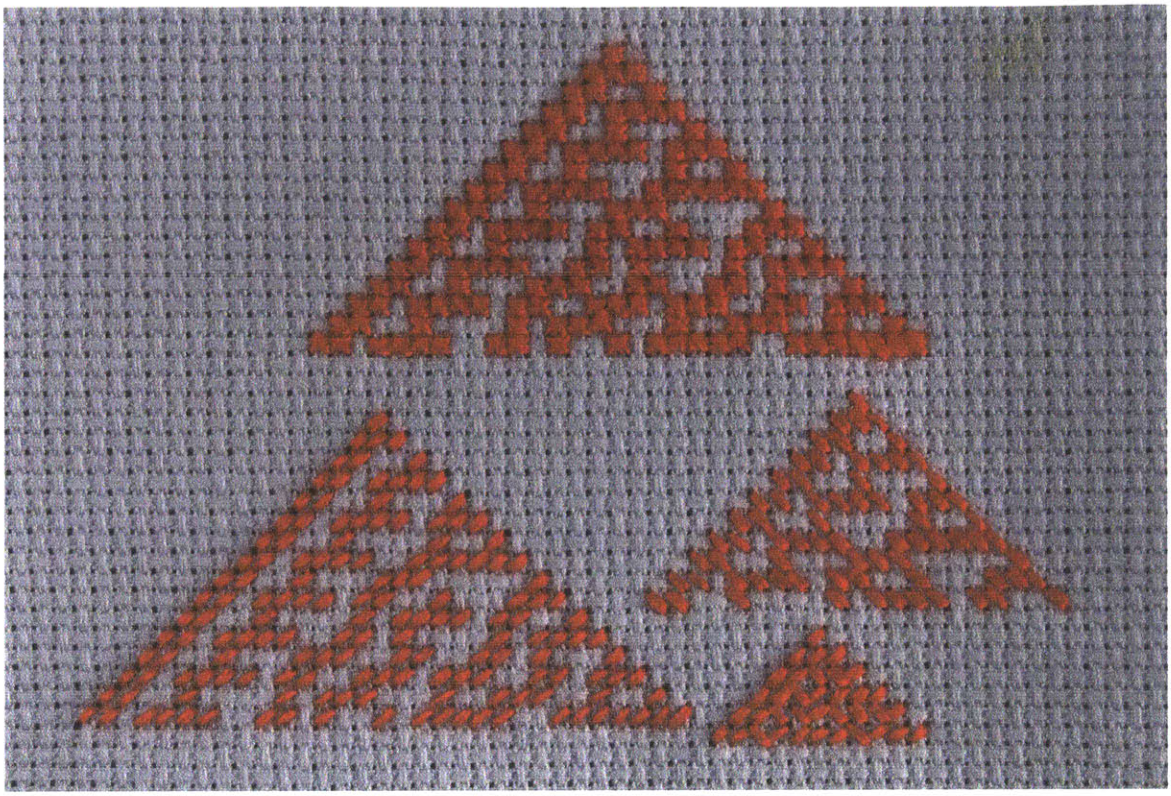


Figure 20: Same rule, different effects.†

Part II

A Di_Alogue Logic

Concepts , et al.

Up until this point, I have examined examples and exercises that do not exhibit true di_alogue nature. In this section, I will look at true di_alogue practices, as well as, in the midst of this, look at some interesting concepts in computer science that relate to the combination of the digital and the physical/analogue worlds.

To reiterate from before: Di_Alogue practices, processes, et al., exhibit a completely intertwined combination of digital concepts and practices with analogue materials, crafts, and/or processes. In di_alogue systems, the intrinsic logic of the analogue, be it from its geometry, materiality, of the conventions related to it, become active members in computational practice, yet they do so without kneeling to conventional computational norms; instead, a new logic, a Di_Alogue Logic is formed anew.

This concept is, albeit, nebulous; it is meant to be. In order to keep it open and to request future work and additions to the documentation of this "world," I have tried to define "di_alogue" in a loose way with the backing of examples of such processes, as well as processes that are similar, yet not quite "there" (see the previous section, "Printing with Craft").

Logic Matter

One of the only examples that I have been able to find which really engages a traditional craft, that of building, with digital logic, to create something unique and different from its constituent digital and analogue parts, is the masters thesis work of Skylar Tibbits, of the Massachusetts In-

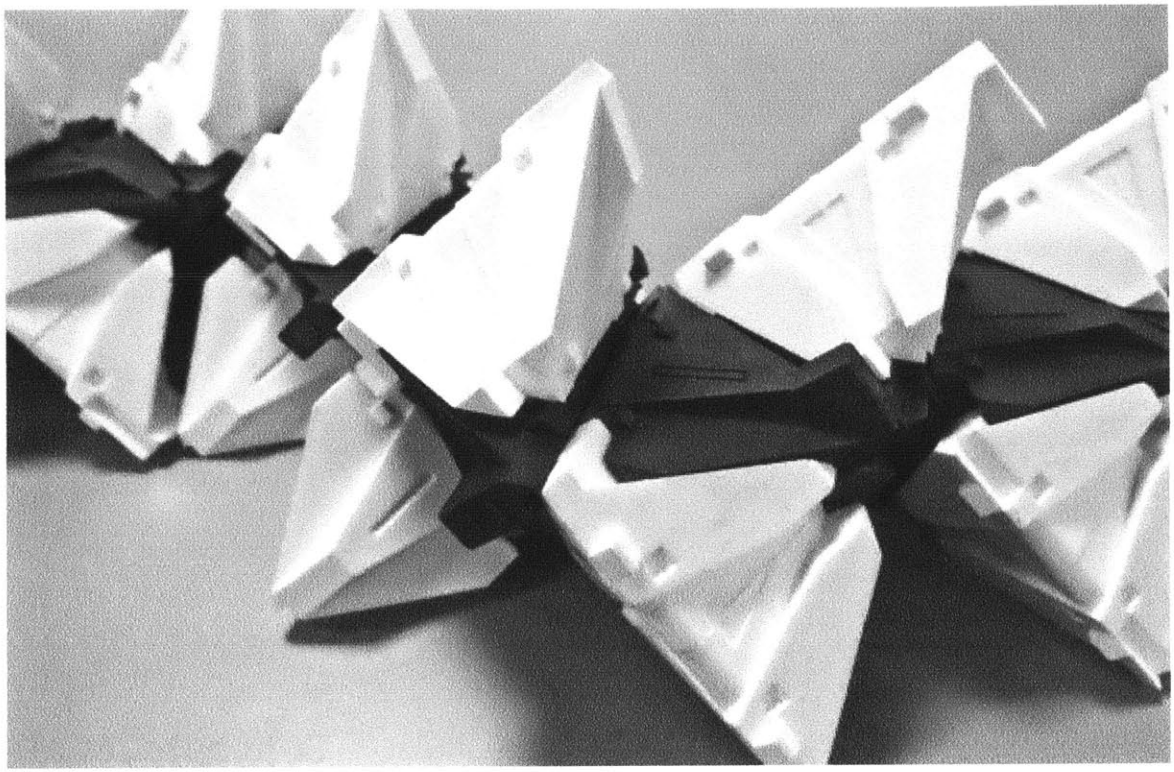


Figure 21: Logic Matter.†

stitute of Technology (MIT), For his thesis, *Logic Matter: Digital logic as heuristics for physical self-guided-assembly*, Tibbits creates a building system in which the geometry, an, thus, an intrinsic, physical logic, allow for the "builder" to perform "computations" with the material itself.[1] This material acts as a NAND (or a Negating AND gate) and can take inputs and give outputs (with user participation as well as interpretation).

Tibbits's blocks are not the standard, box forms that we are used to. Instead they are simple polyhedron, whose faces relate to either a TRUE or FALSE input or a TRUE or FALSE output. Additionally, the individual building blocks contain holes which become filled (or not) depending on inputs from other blocks. The state of these holes then dictate how the building, or computation, can proceed bu blocking access points to the already-built block structure.

On one level this is just a change in the notation of standard, symbolic-based logic. Rather than symbolic operators, such as "AND" or "¬", Tibbits uses geometric (ie. three dimensional) symbols, so in a way he is "printing" using a new media, new representation. Yet, Tibbits skirts purely printing "old notion" of logic by relying on the geometry to define the logic itself. His Logic Matter's unique form informs the way in which the logic works, so, in a way, it has its own, intrinsic logic. This project presents one of the closest

1 Tibbits, Skylar. Logic Matter: Digital logic as heuristics for physical self-guided-assembly. MA thesis. Massachusetts Institute of Technology, 2010. Print.

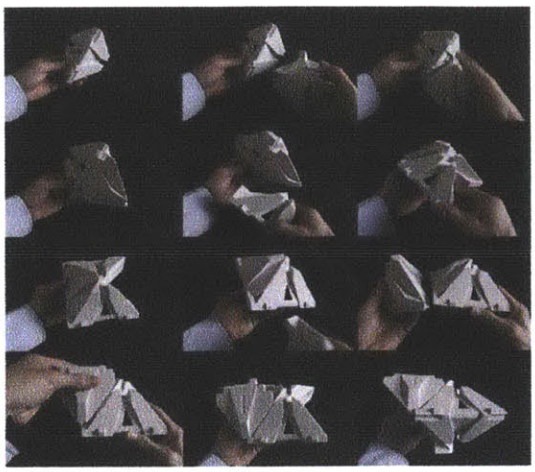


Figure 61. 4 Logic Matter modules demonstrating programmability of base configuration [0,0] = 1. The final configuration demonstrates a 180 degree vertical rotational difference from output = 0.

$$[0,0] = 1$$

Figure 22: NAND gate: [0, 0]=1.†

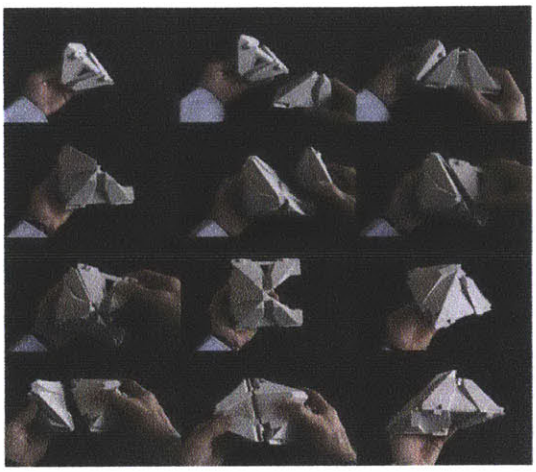


Figure 62. 4 Logic Matter modules demonstrating programmability of base configuration [1,1] = 0. The final configuration demonstrates a 180 degree vertical rotational difference from output = 1.

$$[1,1] = 0$$

Figure 23: NAND gate: [1, 1]=0.†

manifestations of a Di_Alogue Craft as has been created.

Conservative Logic

From further study beyond the superficial outlines provided by Tibbits, it is clear that if a di_alogue craft, as well as a corresponding di_alogue logic, it will need to rely on previous, though abstract, research in the field of computer science. As stated previously, the seminal paper which laid down the foundations for the computing age are of particular interest.

One sub-field which I have come across concerns computing systems which turn away form the orthodox, boolean logic which runs our computing systems today. The areas of "reversible" or "conservative logic" as lined out by Edward Fredkin and Tommaso Toffoli in their seminal paper, "Conservative Logic" are of extreme interest as they offer computing solutions specifically meant for the physical world.[1]

Essentially, conservative logic is a logic system *similar* to the classic Boolean (0/1) logic system that reigns over computation today. Conservative logic, however, conserves bits of information (generally the '1' in computational notation) so as to "reflect in ins axioms certain fundamental principles of physics."[2] So, while in a current AND gate, the inputs [1, 0] would output a [0] (True AND False is False), using a conservative gate, such as the Fredkin Gate, the inputs [0, 1, 0] would output [0, 0, 1] (see Figure 24).[3] While the two functions look different, they are performing the same function via the use of "garbage" vari-

1 Fredkin, Edward & Tommaso Toffoli. *Conservative Logic. International Journal of Theoretical Physics* (Vol.21, Nos. 3/4, 1982). Print. 219 - 253.

2 ibid., 219.

3 ibid., 227.

u	x_1	x_2		v	y_1	y_2
0	0	0		0	0	0
0	0	1		0	1	0
0	1	0		0	0	1
0	1	1	→	0	1	1
1	0	0		1	0	0
1	0	1		1	0	1
1	1	0		1	1	0
1	1	1		1	1	1

Figure 24: The Fredkin Gate.†

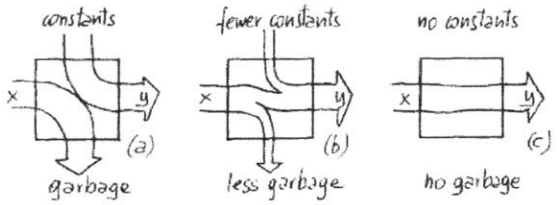


Figure 25: Use of "constants" and "garbage" in conservative logic.†

ables (see Figure 25).[1]

Overall, the orthodox and conservative logic systems do not seem all that different, yet, "the central result of conservative logic is that *it is ideally possible to build sequential circuits with zero power dispersion*," a fact that is not true of our current computing paradigm.[2]

This fact, that such conservative logic systems, such as that proposed by the Fredkin Gate, conserve energy/material/information suggests that they have a unique ability to "talk" with craft as traditional craft often has "rules" concerning the adding or subtracting of material. For example, in embroidery, the crafter only uses one thread at one time.

To show the veracity of their concept, Fredkin and Toffoli created a theoretical catalogue of "Billiard Ball Logic Gates."[3] This system, assuming a frictionless world, was able to "compute" using collision based gates. Balls (with no elastic properties), representing bits of information, could collide with walls or each other to give outputs to various functions.

One pair of researchers that have heavily contributed to the field of conservative logic are Tsuto-

1 ibid., 230.

2 ibid., 227.

3 ibid., 229.

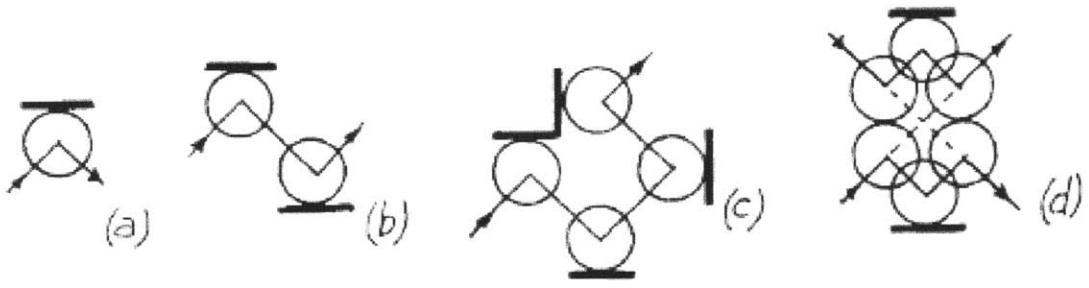


Figure 26: Billiard Ball logic gates.†

TABLE I
MINIMAL 3-3 CLC's

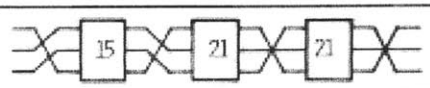
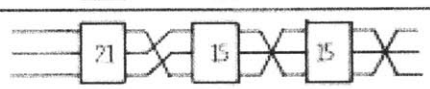
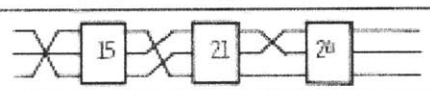
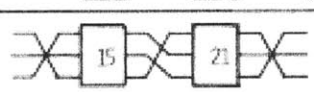
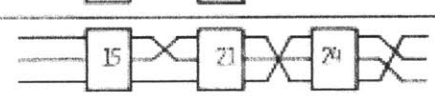
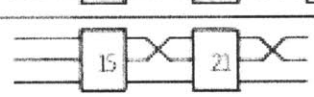
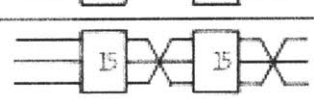
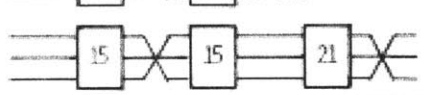
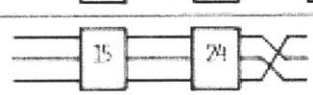
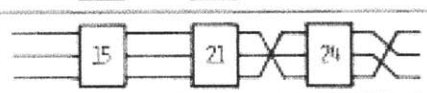
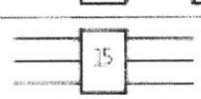
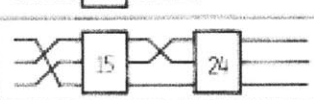
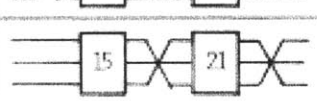
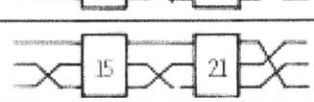
#	$\alpha^{\#}$	α	Boolean expression	$\phi_1\phi_2\phi_3$	Minimal circuit
1	1	(0 3 3) (2 2 2)		3 3 0	
2	3	(1 3 3) (2 2 2)		3 3 0	
4	18	(2 2 2) (1 1 2)		3 1 1	
5	5	(2 3 3) (1 1 2)		2 1 3	
6	19	(3 1 1) (2 1 2)		1 1 1	
7	11	(2 2 3) (1 1 2)		1 1 0	
8	8	(2 2 0) (1 1 3)		2 1 1	
9	16	(2 2 0) (1 2 3)		3 0 1	
10	28	(2 2 2) (1 1 3)		2 1 2	
12	20	(1 1 2) (2 1 3)		1 0 1	
13	13	(1 1 2) (2 1 3)		1 1 2	
15	21	(2 2 3) (1 2 3)		3 0 2	
16	30	(1 2 2) (2 1 3)		2 2 0	
17	17	(2 2 3) (1 1 2)		1 1 1	
22	22	(2 1 2) (1 1 3)		1 1 2	
23	23	(1 2 3) (1 2 3)		0 0 3	

Figure 27: 3-input, 3-output cascade functions.†

mu Sasao and Kozo Kinoshita (see Bibliography for a list of works). Sasao and Kinoshita were able to show conservative logic's universality (in the computational sense),[1] as well as to address several configuration questions.[2,3,4] In one paper, Sasao and Kinoshita produced wonderful, woven paths to show the unique number of cascade functions using conservative logic gates (see Figure 27).[5]

1 Sasao, Tsutomu & Kozo Kinoshita. "Conservative Logic Elements and Their Universality." *IEEE Transactions on Computers*. Vol. c-28, No. 9, 682-685. (1979). Print.

2 ----. "Cascade Realization of 3-Input 3-Output Conservative Logic Circuits." *IEEE Transactions on Computers*. Vol. c-27, No. 3, 214-221. (1978). Print.

3 ----. "On the Number of Fanout-Free Functions and Unate Cascade Functions." *IEEE Transactions on Computers*. Vol. c-27, No. 1, 66-72. (1979). Print

4 ----.. "Realization of Minimum Circuits with Two-Input Conservative Logic Elements." *IEEE Transactions on Computers*. Vol. c-27, No. 8, 749-752. (1978). Print

5 ----. "Cascade Realization of 3-Input 3-Output Conservative Logic Circuits." *IEEE Transactions on Computers*. Vol. c-27, No. 3, 214-221. (1978). Print.

A Di_Alogue Cross-Stitch

Wanting to try my hand at developing my own truly Di_Alogue interpretation of cross-stitch/ half-cross-stitch, I proceeded with various experiments with embroidery floss until I produced a heuristic algorithm which both respects cross-stitch convention while being able to preform logical operations.

Before this could begin, I had to understand the constituent parts that make up the cross-stitching system.

A Critical Look at Cross-Stitch

The basics of cross-stitch and half-cross-stitch were quickly covered in Section: Printing with Cross-Stitch.

On further examination, cross-stitch can be cleaved into two different processes: threading which lays embroidery floss on the "front" of the fabric/work and in a diagonal relation to the grid defined by said fabric/base material, and threading which lays the floss on the "back" and along the major axis directions defined by the fabric/ base material (see Figure 28). While the "front-work," or the diagonal stitches, form the desired pattern or image that will be viewed after the work is done, the "back-work", or the horizontal or vertical stitches, form the backbone of th process which correctly positions the front-work on the fabric/base material (hereafter referred to as "the Medium").

It should be noted that in the process of cross-stitching a front-work stitch is always followed by a back-work stitch, and a back-work

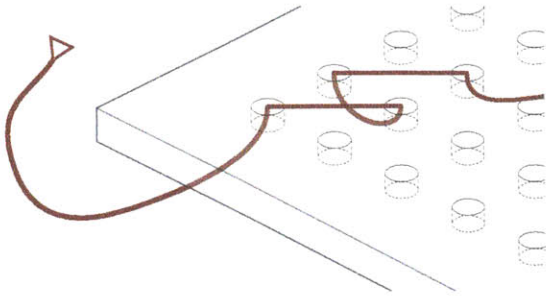


Figure 28: Cross-stitch/half-cross-stitch.†

stitch is always followed by a front-work stitch. This is to maintain the stability of the pattern and properly affix it to the medium. This will be referred to as the Front-Back-Front-Back Rule.

If we consider the act of "reading" and consuming a work of cross-stitch, it is the front-work which we interact with, thus it would make sense to, in the translation of cross-stitch into a Di_Alogue craft, make the front-work relate to inputs and outputs to logic functions. As logic gates actually do the manipulation of the variables, it is quite clear that in such a di_alogue craft that the back-work, which manipulates the location of the floss, should act, in some way to the logic gate used in the manipulation of the cross-stitched inputs/outputs.

In a simple two-input, one-output logic function, there exists four entities: two arguments (inputs), one operand (gate) which acts upon these arguments, and one value (output). Because of the Front-Back-Front-Back Rule, if we wish to chain the fewest number stitches together to realize a logic gate, the complete logical computation will take five stitches. The first, third, and fifth will be front stitches and will relate to the two input stitches and one output stitch, respectively. This leaves two back-work stitches, which, in the schema described above, need to relate to the operand of the function, such as the AND, OR, NOT, NAND, logical functions.

If we look at these first five stitches, what are the possible ways that they can be applied to the medium? The front-work stitches can travel in all of the diagonal directions (4: Up-and to the Right, Up-and-to-the-Left, Down-and-to-the-Right, Down-and-to-the-Left) from their initial point, and the back-work stitches can travel in all ordinal/axial directions (4: Up, Down, Left, Right).

Form here, we can imagine what such a computation will look like. By inputing four stitches: input one (front-work), operand part-one (back-work), input two (front-work), operand part-two (back-work), the system puts "the stitcher" in position to make the last front-work stitch. This stitch could be chosen based on a heuristic, that is determined or attached to the particular operand pair, that would determine which of the four options to choose. This way this last stitch would be deterministic based on the choice of individual inputs as well as the choice of operand-stitch-pair (the operand as a whole). Because the value of the last stitch is in relation to the operand and the input stitches, the heuristic used to determine the value must relate to these stitches as well. Since the positioning of all the subsequent stitches relate to the first stitch, this heuristic must relate to the first stitch. From practice, it makes sense to craft this heuristic around motion (in the value stitch) toward or away from a horizontal and/or vertical imaginary line projecting from the center of the first, input stitch.

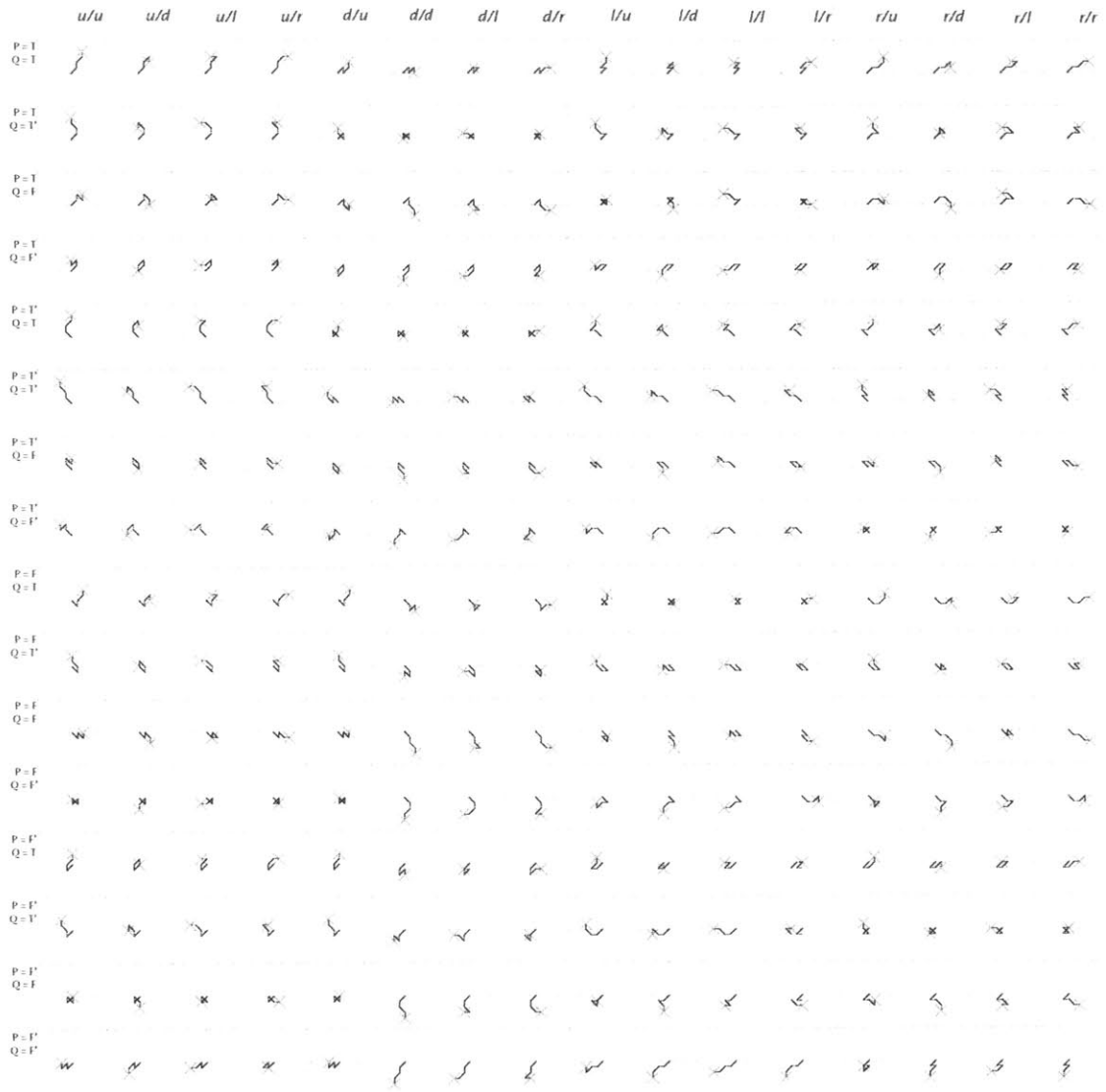


Figure 29: All possible combinations and permutations of first five "gestures" in half-cross-stitch.†

In total there will be:

```
>>> 4 × 4 × 4 × 4 × 4 = 1024
```

number of possible combinations of these five stitches. (16 possible input pairs, 16 possible operand pairs, and 4 possible value stitches.)

Figure 29 displays all of the possible stitch combinations though the fourth stitch in the sequence (with the four value stitches displayed in grey, at the end). For these calculations, I used the following notation system:

```
>>> P and Q represent the two input
    stitches.
>>> P and Q can be T or T' or F or F', these
    relate to stitches that lay floss down by
    "moving" Up-and-to-the-Right or Up-and-
    to-the-Left or Down-and-to-the-Right or
    Down-and-to-the-Left, respectively.
```

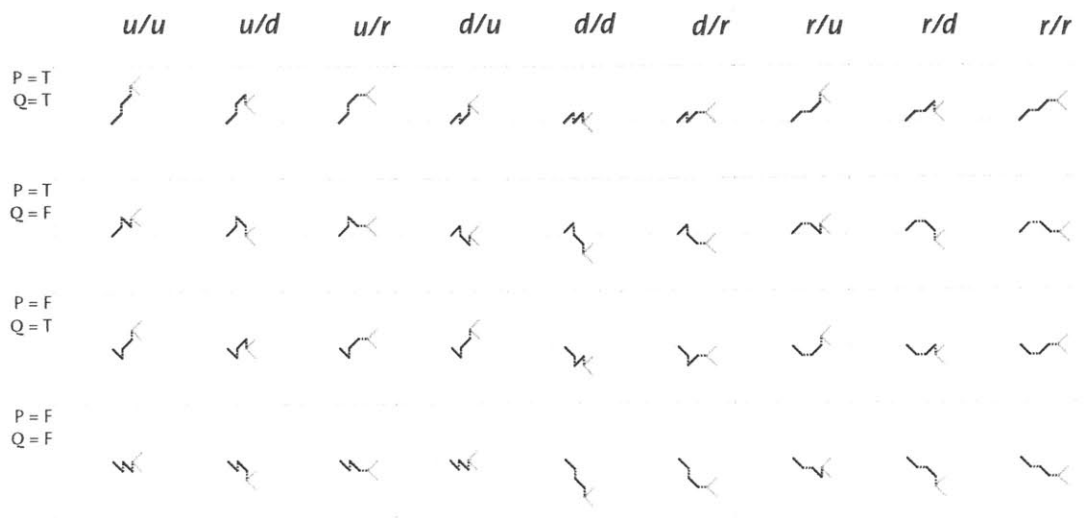


Figure 30: Combinations and permutations that follow convention.†

```
>>> The back-work stitch pairs are
    represented as [stitch-2-of5]/[stitch-4-
    of-five]
>>> The back-work stitches proceed u or d
    or l or r, these relate to up or down
    or left or right, respectively. So, an
    operand pair in which the first stitch
    "moves" down, and the second "moves" the
    floss to the right would be 'd/r.'
```

Obviously, all of these combinations would not be allowed in traditional cross stitch, thus, by removing any motion to the left of the work (cross-stitch usually moves from the left *to* the right), we are left with 36 "legitimate" input/output and operand combinations: nine "operand pairs" with different notation, and four different input/output pairs. And, since the last stitch will move either Up-and-to the-Right (as in previous sections, this will be a TRUE Stitch) or Down-and-to-the-Right (as in previous sections, this will be a FALSE Stitch), the heuristic only has to concern the imaginary horizontal line that projects from the mid-point of the first, input stitch. Thus, the last stitch, the value stitch, would be guided by the input stitches as well as the operand pair and it's attached heuristic to go either "towards the line" or "away form the line."

Through analysis, each of these operand pairs, when used with the four different input/output pair will relate to either a logic function or will be "equivalent to" another of the operand pairs.

In oder from left to right along the choices of operand pair and heuristic ("toward" or "away") in Figure 30, the following conclusions can be made. No matter the input, a particular operand pair and a particular heuristic will relate to the following logic function or truth value or unique argument (argument one is 'P,' argument two is 'Q'):

```
>>> u/u - away: always equals TRUE
>>> u/u - towards: always equals FALSE
```

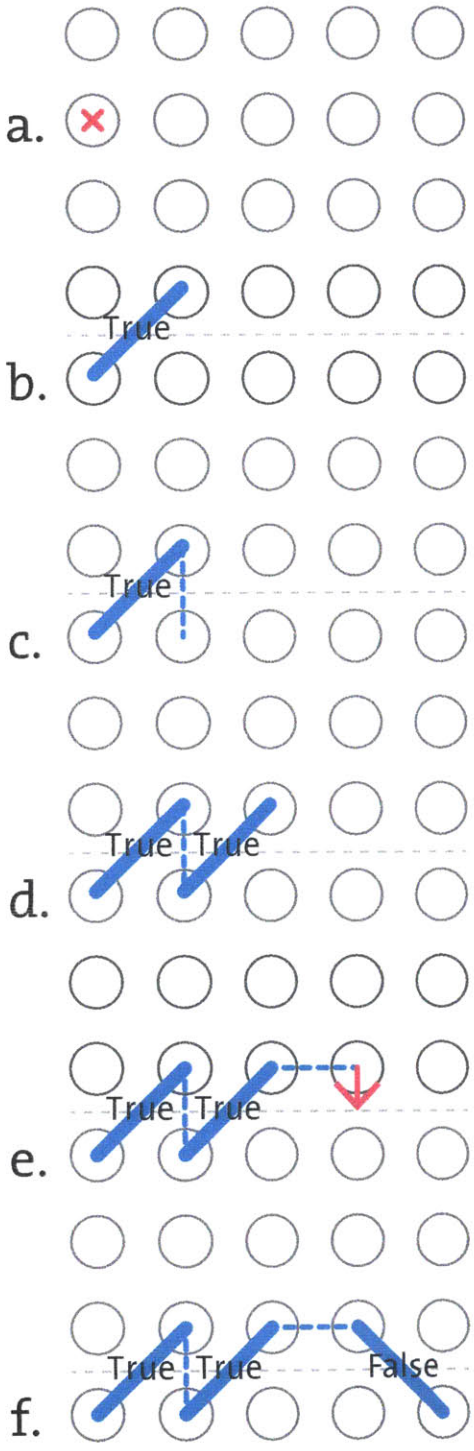


Figure 31: The NAND Stitch heuristic algorithm.†

```
>>> u/d - away:  Q
>>> u/d - towards:  NOT Q
>>> u/r - away:  P OR Q
>>> u/r - towards:  P NOR Q
>>> d/u - away:  equivalent to u/d - away
>>> d/u - towards:  equivalent to u/d -
    towards
>>> d/d - away:  always equals FALSE
>>> d/d - towards:  always equals TRUE
>>> d/r - away:   P AND Q
>>> d/r - towards:  P NAND Q
>>> r/u - away:  equivalent to u/r - away
>>> r/u - towards:  equivalent to u/r -
    towards
>>> r/d - away:  equivalent to d/r - away
>>> r/d - towards:  equivalent to d/r -
    towards
>>> r/r - away:  equivalent to u/d - away
>>> r/r - towards:  equivalent to u/d -
    towards
```

From here-on-out, I will mostly be considering the operand pair and heuristic: d/r - towards, this performs an NAND function upon the chosen inputs.

NAND Gate Stitch

The NAND Stitch relates to the NAND Gate/Logic function, which is a *universal* logic functions, that means that it can be combined with of the NAND Gates to make any other type of logic function.

Figure 31 describes the stitching of the NAND Stitch with an input pair of: TRUE, TRUE.

```
>>> (a)  A starting point is chosen in the
    medium (the needle will start in the
    back and come through the hole baked by
    the magenta 'x' to initiate the stitch).
>>> (b)  A TRUE Stitch is made (needle now in
    back of the medium).
>>> (c)  The first part of the operand pair (a
    down stitch) is made.
>>> (d)  The second TRUE Stitch is 'entered.'
>>> (e)  The second part of the operand
    pair (a right stitch) is made.  The
    needle is now in front of the medium
    and the stitcher now must decide where
    to proceed to make the last stitch,
    the value stitch.  Since the imaginary
    horizontal line that projects from the
    mid-point of the first stitch is below
    where the floss is now exiting the
    medium, the next stitch will proceed
    towards the line by 'moving' Down-and-
```

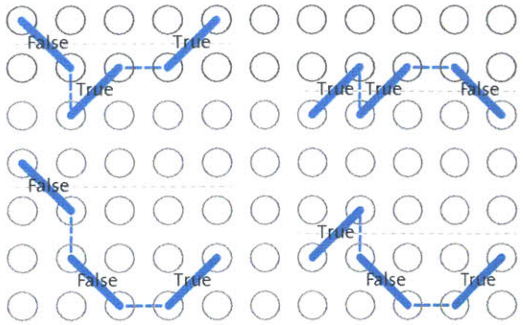


Figure 32: The NAND Stitch with all possible (and conventional) inputs.†

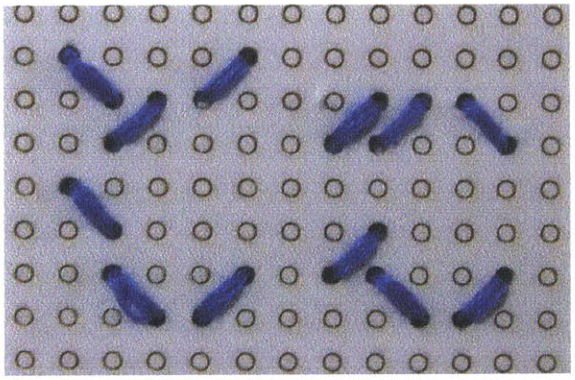


Figure 33: The NAND Stitch with all possible (and conventional) inputs in actuality.†

```
    to-the right.
>>> (f)  The result (a FALSE Stitch) is made
    in accordance with the logic in step (e).
```

Figure 32 draws out the resulting computation when all input pair choices are used; Figure 33, depicts the actual half-cross-stitched result.

By chaining together many of these operations, a stitcher can make patterned stitched chains. To do this, they only need to use the value stitch of the last computation as the input of the next; this process is identical to that which Tibbits used.[1] Figure 34 shows a chain computed using the input string (TRUE == 1, FALSE == 0): 1011111... And in Figure 35, a random chain is created using

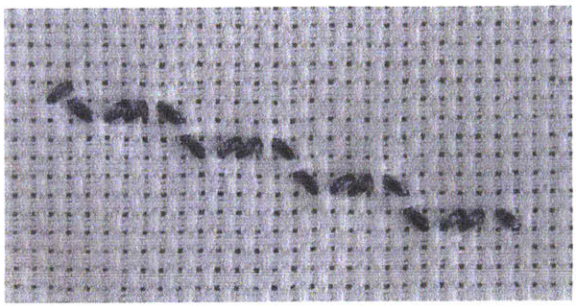


Figure 34: An NAND Stitch computation started with an input pair: TRUE, FALSE, and continued with TRUE Stitches.†

a string of random truth values. This sudo-random string is:

```
>>> 01000011000101100110.
```

Figure 35, 36, 37, and 38, are all NAND Stitch chains where the firs two input pairs are one of the four unique input pairs (TRUE, TRUE; TRUE, FALSE; FALSE, TRUE; FALSE, FALSE). The chain is continued by taking the value stitch of the last pair and computing the NAND function with a stitch of the opposite value.

Figure 40 and Figure 41, go back and examine the various equivalences that occur between the operand pairs (along with their heuristics). Though they start off looking different at the beginning of each chain's computation, they end up reaching the same pattern in the end. This behavior is very similar to the local chaos, but long-distance order of cellular automata. Please see the captions for more detail.

1 Tibbits, 67.

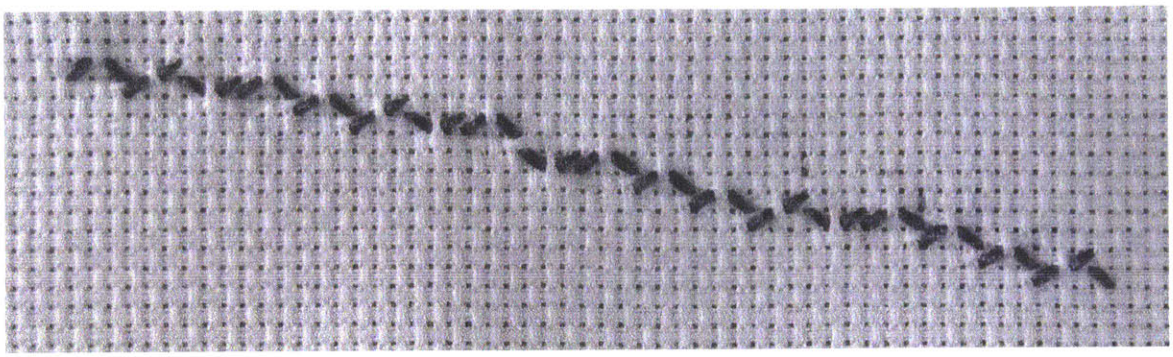


Figure 34: NAND Stitch chain with random string of inputs.†

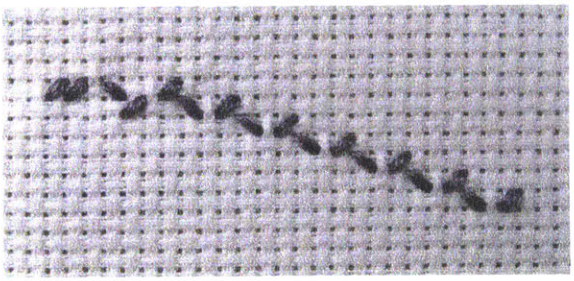


Figure 35: An NAND Stitch computation started with an input pair: TRUE, TRUE, and continued with the opposite of whatever the last value stitch was.†

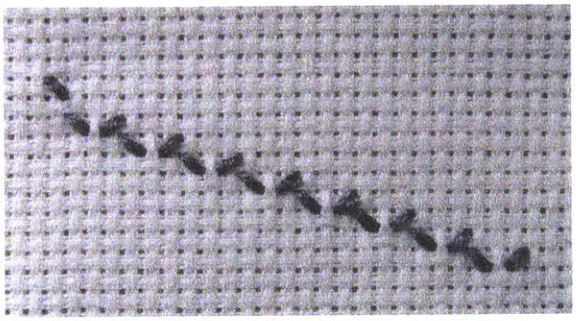


Figure 37: An NAND Stitch computation started with an input pair: FLASE, FALSE, and continued with the opposite of whatever the last value stitch was.†

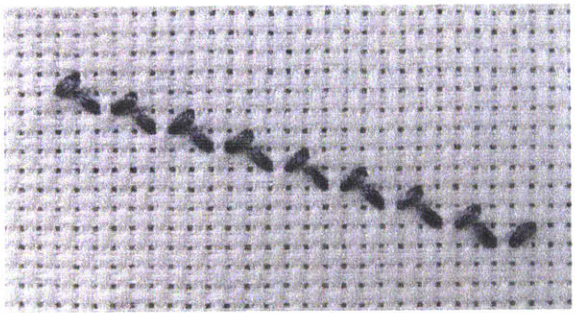


Figure 36: An NAND Stitch computation started with an input pair: TRUE, FALSE, and continued with the opposite of whatever the last value stitch was.†

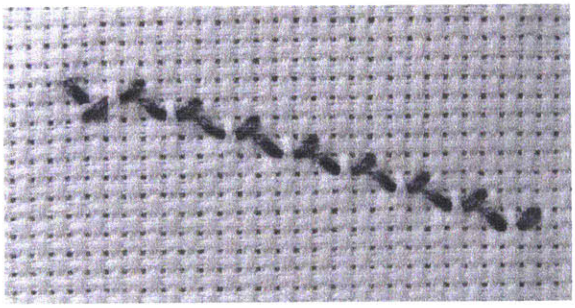


Figure 38: An NAND Stitch computation started with an input pair: FALSE, TRUE, and continued with the opposite of whatever the last value stitch was.†

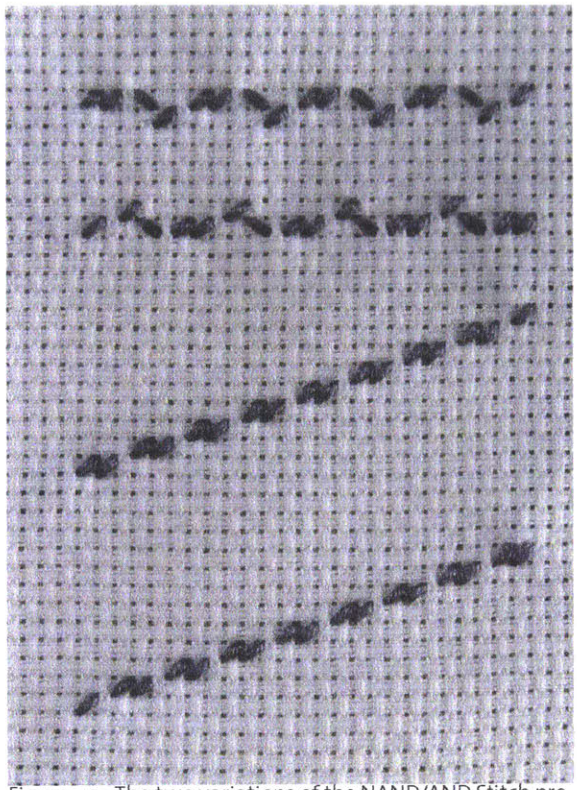


Figure 40: The two variations of the NAND/AND Stitch predicted by Figure 31. Each depicts a chain of TRUE values. From the top, each chain relates to: d/r - towards (NAND); r/d - towards (NAND); d/r - away (AND); r/d - away (AND) .†

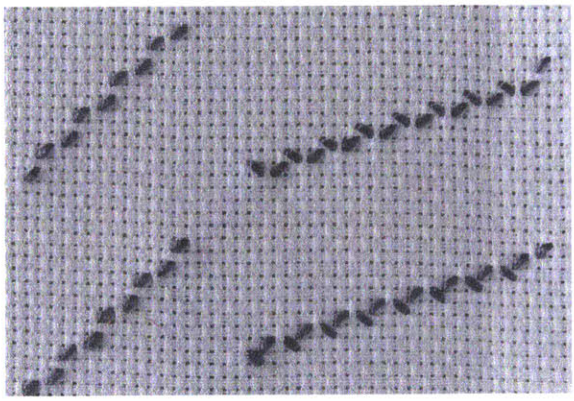


Figure 41: The two variations of the OR/NOR Stitches predicted by Figure 31. Each depicts a chain of TRUE values. From the top left and moving counter-clockwise, each chain relates to: u/r - towards (OR); r/u - towards (OR); u/r - away (NOR); r/u - away (NOR) .†

Conclusion & Acknowledgments

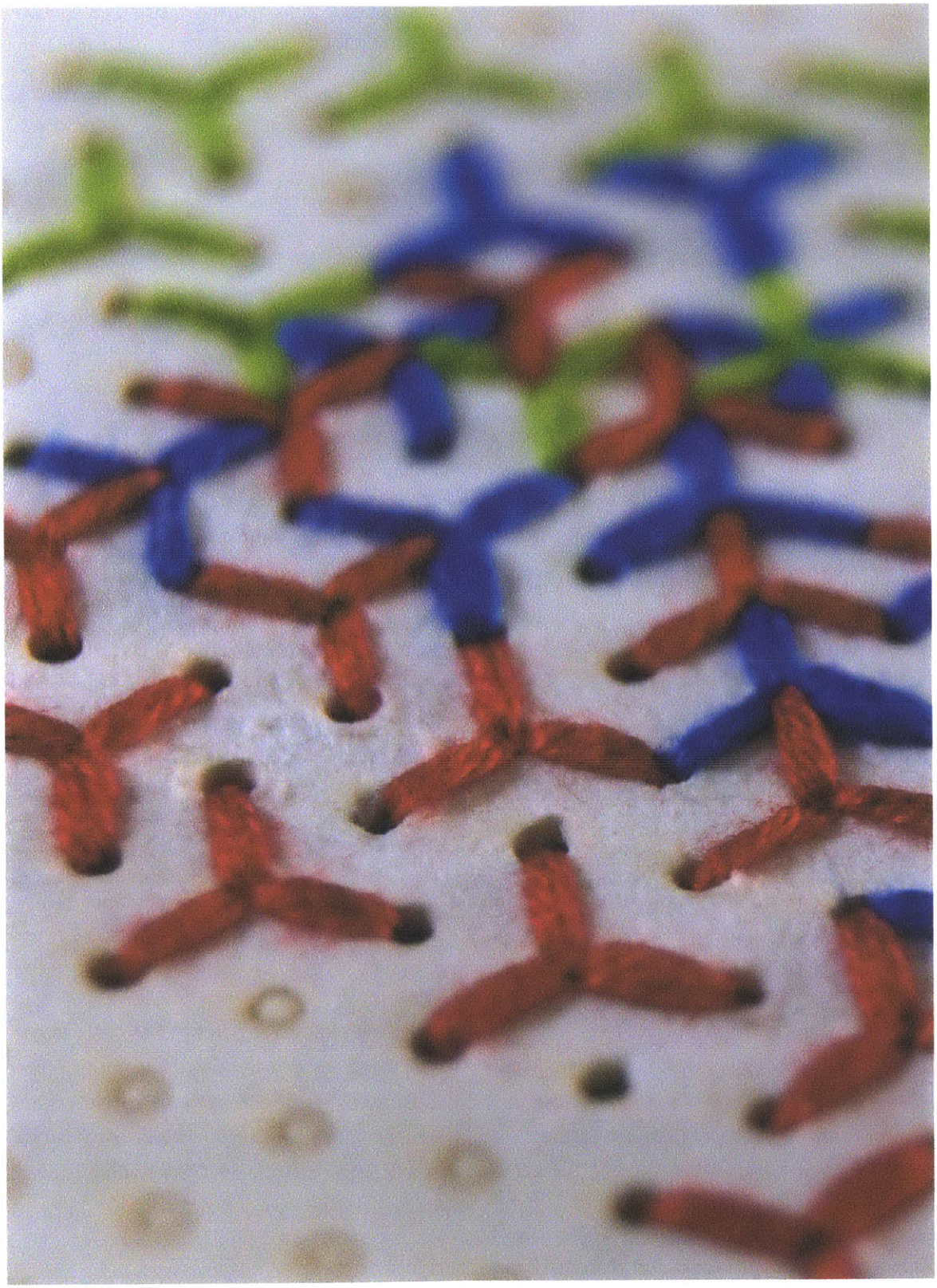
The Di_Alogue World has yet to be mapped, a few have ventured into it , see what it is like. The goal of this thesis was to make such an excursion. I believe this area of research, into di_alogue practices and the like will be very valuable in the increasingly interdisciplinary world in which we find ourselves. Even more so, I believe that by pushing the boundaries on conventional notions that we can begin, in the words of Bill Hubbard Jr., "re-see" our world, and therefore, re-define it and make it ours.

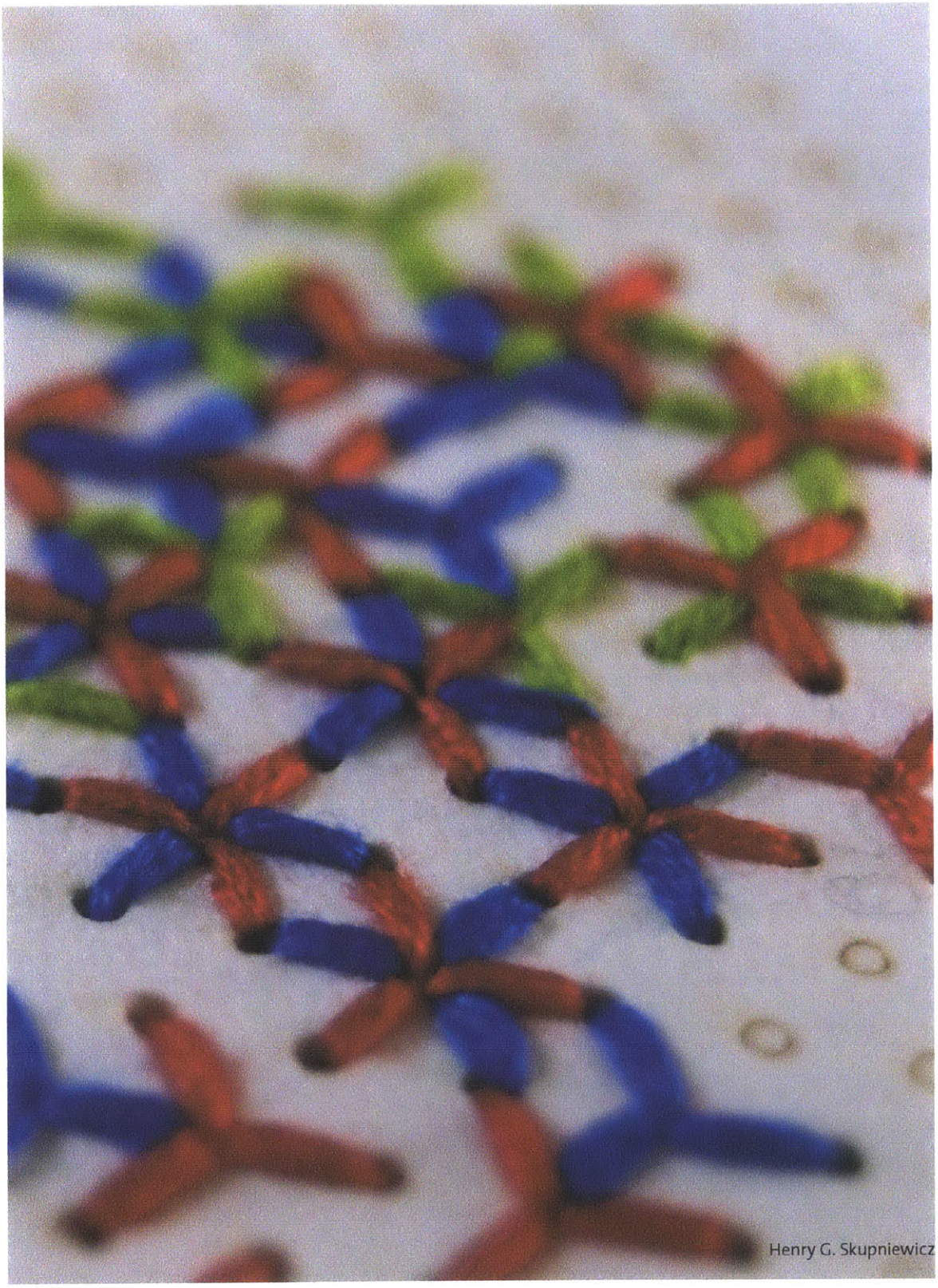
I would like to thank many for the help that I have received on this thesis as well as over the past four years on my undergraduate education.

I would like to thank my classmates and friends, especially those on the Light Weight Men's Crew, and all fellow residents (past and present) of Burton Third. I would like to thank the faculty and staff of the MIT Architecture Department (especially George Stiny) for pushing me into new areas by putting me through hell during my first few studios; with out this experience I would have never realized my calling to teach and to research. I need to thank the Institute, herself, for showing me how much I do, but, mostly, how much a *don not* know, thus setting me up for a life of learning. Tech, I am sure our paths will cross again.

I owe everything to my parents, Gail and George, for allowing, in so many ways, for me to go on this crazy four-year journey at MIT. And I thank my sister, 'ERPS,' for keeping a wry smile on my face the whole time. I am truly honored to call you my own; I love you.

Henry George Skupniewicz




Henry G. Skupniewicz

Bibliography

Image citations appera in the following section.

Arkin, Esther M., et al.

"The Embroidery Problem." *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG2008).* Montéal, Québec, August 135-138, 2008. Print

Ashley, Clifford W..

The Ashley Book of Knots. 1944. Ed. Geoffrey Budworth. Boston: Faber and Faber, 1993. Print.

Belcastro, Sarah-Marie & Carolyn Yackel, eds..

Making Mathematics with needlework: Ten Papers and Ten Projects. Natick, MA: A K Peters, Ltd., 2008. Print.

Biedl, Therese, et al..

"Cross-Stitching Using Little Thread." *Proceedings of the 17th Canadian Conference on Computational Geometry (CCCG'05).* 199-202. (2005). Print.

Bruce, J.W., et al..

"Efficient Adder Circuits Based on a Conservative Reversible Logic Gates." *Proceedings of VLSI, 2002.:IEEE Computer Society Annual Symposium.* 26 April, 2002: Pittsburgh, PA. 74-79. Print.

Burks, Arthur W., ed..

Essays on CellularAutomata. Chicago: University of Illinois Press, 1970. Print.

Dietz, Ada K..

Algebraic Expressions in Handwoven Textiles. Ed. Robert Kirkpatrick. Louisville, KY: Little Loomhouse, 1949. Web PDF. 23 Nov, 2012. Print.

de Dillmont, Th..

The complete Encyclopedia of Needlework. Philadelphia: Running Press, 1974. Print.

Engeler, Erwin.

Introduction to the Theory of Computation. New York: Academic Press, 1973. Print.

Essinger, James.

Jacquard's Web: How a hand loom led to the birth of the information age. Oxford: Oxford University Press, 2004. Print.

Fredkin, Edward & Tommaso Toffoli.

"Conservative Logic." *International Journal of Theoretical Physics.* Vol.21, Nos. 3/4, 219-253. (1982). Print.

Griswold, Prof. Ralph E.

Mathematical and Computational Topics in Weaving. Self-published, 2006. Web PDF. 23 Nov, 2012. Print.

Held, Shirley E..

Weaving: A Handbook of the Fiber Arts. 2nd ed.. New York: Holt, Rinehart and Winston, 1978. Print.

Lee, S. Y. & H. Chang.

"Magnetic Bubble Logic." *IEEE Transaction on Magnetics.* Vol. MAG-10, No. 4. 1059-1066. (1974). Print.

Newman, James R..

The World of Mathematics: A small library of the literature of mathematics from A'h-mosé the Scribe to Albert Eistein, presented tih commentaries and notes. New York: Simon Schuster, Inc., 1956. Print.

Penrose, L.S..

"Self-Reproducing Machines." *Scientific American.* Vol. 200, No. 6, 105-114. (1959). Print.

Perkowski, Marek, et al..

"Regularity and Symmetry as a Base for Efficient Realization of Reversible Logic Circuits."

Proceedings of IWLS 2001. 90 - 95. (2001). Print.

Sasao, Tsutomu & Kozo Kinoshita.

"Cascade Realization of 3-Input 3-Output Conservative Logic Circuits." *IEEE Transactions on Computers*. Vol. c-27, No. 3, 214-221. (1978). Print.

"Conservative Logic Elements and Their Universality." *IEEE Transactions on Computers*. Vol. c-28, No. 9, 682-685. (1979). Print.

"On the Number of Fanout-Free Functions and Unate Cascade Functions." *IEEE Transactions on Computers*. Vol. c-27, No. 1, 66-72. (1979). Print

"Realization of Minimum Circuits with Two-Input Conservative Logic Elements." *IEEE Transactions on Computers*. Vol. c-27, No. 8, 749-752. (1978). Print

Shannon, Claude E..

"Communication in the Presence of Noise." *Proceedings of the I.R.E.*. Vol. 37, Issue 1, 10-21. (1949). Print.

"Communication Theory - Exposition of Fundamentals." *Information Theory, Transactions of the IRE Professional Group*. Vol.1, No. 1, 44-47. (1953). Print.

"Computers and Automata." *Proceedings of the I.R.E.*. Vol. 44, Issue 10, 1234-1241. (1953). Print.

"General Treatment of the Problem of Coding." *Information Theory, Transactions of the IRE Professional Group*. Vol.1, No. 1, 102-104. (1953). Print.

Spuybroek, Lars, ed..

Textile Tectonics. Rotterdam: NAi Publishers, 2011. Print.

Tibbits, Skylar.

Logic Matter: Digital logic as heuristics for physical self-guided-assembly. MA thesis. Massachusetts Institute of Technology, 2010. Print.

Toffoli, Tommaso.

"Bicontinuous Extentions of Invertible Combinatorial Functions." *Math. Systems Theory*. Vol. 14, 13-23. (1981). Print.

"Computation and Construction Universality of Reversible Cellular Automata." *Journal of Computer and System Sciences*. Vol. 15, 213-231. (1977). Print.

"Reversible Computing." MIT Laboratory for Computer Science. 1980. Print.

"Reversible Computing" (abridged version). MIT Laboratory for Computer Science. 1980. Print.

von Neumann, John.

Theory of Self-Reproducing Automata. Editied and completeed by Burks, ArthurW.. London: University of Illinois Press, 1966. Print.

Yanushkevich, Svetlana N. & Vlad P. Shmerko.

Introduction to Logic Design. New York: CRC Press, 2008. Print.

Figure Index

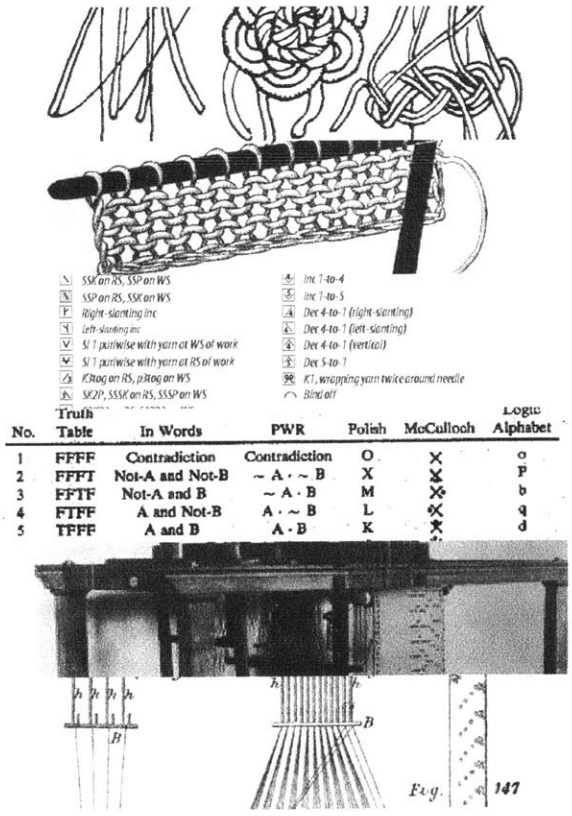


Figure 0
"Decorative knotwork from Clifford W. Ashley's *The Ashley Book of Knots*, originally published in 1944."
Ashley, 172.

Figure 1
"A page from *A Text-Book of Needlework, Knitting, and Cutting-Out*, published 1893"

Figure 2
"Symbols used in charted knitting designs."
http://www.craftyarncouncil.com/chart_knit.html

Figure 3
"Traditional logic symbols use in formal logic."
http://jd2718.wordpress.com/2007/05/10/logical-symbols/

Figure 4
"Bouchon's paper-tape guided loom."
Wikimedia Commons.

Figure 5
Bell, T.F. *Jacquard Weaving and Designing*. Google Books.

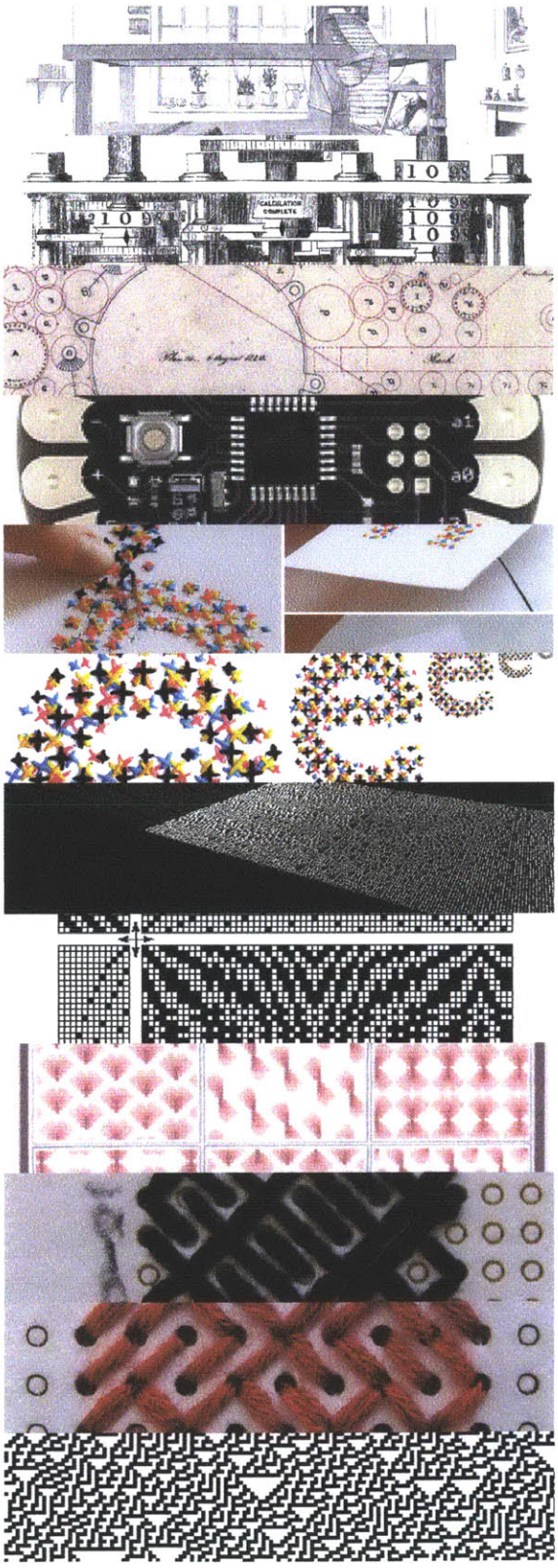


Figure 6
"A Jacquard loom in use."
Bell, T.F. *Jacquard Weaving and Designing.* Google Books.

Figure 6
"Babbage's Difference Engine."
Harper's New Monthly Magazine Volume 0030 Issue 175 (December, 1864). Web. 22 May, 2013.

Figure 8
"Babbage's plan for the Analytical Engine."
Forbes.com. Web. 22 May, 2013.

Figure 9
"LilyPad Arduino"
LilyPad Arduino. Leah Buechley. Web. 14 Dec, 2012.

Figurer 10
"Kasikov in production."
Kasikov, Evilin.

Figure 11
"An "e" in CMKY embroidery."
Kasikov, Evilin.

Figure 12
"Infinite Weft"
Thorp, Jer. Web.

Figure 13
"One of Griswold's designs in standard weaving notation."
Griswold, 172.

Figure 14
"Pattern sampler."
Belcastro, 76.

Figure 15
"The output of Fabric() from [Code 1] stitched into card-stock with cotton embroidery floss."

Figure 16
"Simple binary logic using AND and OR operators represented using embroidery floss."

Figure 17
"'Rule 30.'"
Wikimedia Commons.

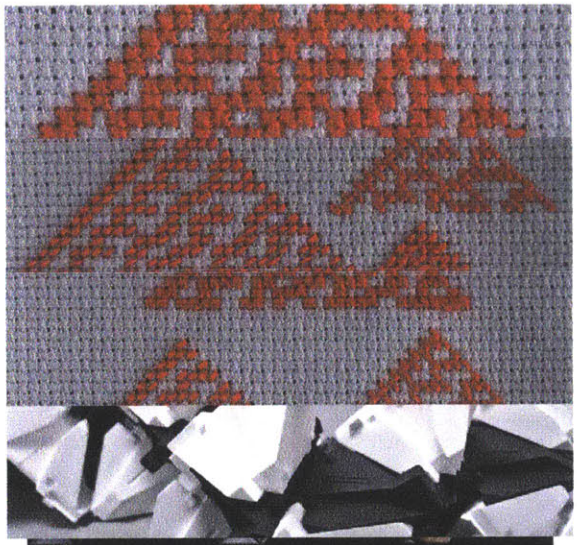


Figure 18
""Rule 30" in traditional cross-stitch."

Figure 19
"Various interpretations of "rule 30" in half-cross-stitch."

Figure 20
"Same rule, different effects."

Figure 21
"Logic Matter."
Tibbits.

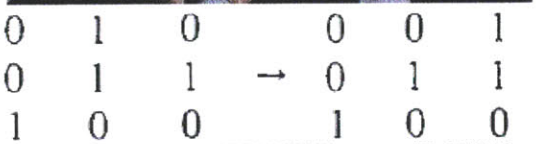


Figure 22
"NAND gate: [1, 1]=0"
Tibbits.

Figure 23
"NAND gate: [1,1]=1"
Tibbits.

Figure 24
"The Fredkin Gate."
Fredkin, 227.

Figure 25
"Use of "constants" and "garbage" in conservative logic."
Fredkin, 250.

Figure 26
"Billiard Ball Gates."
Fredkin, 240.

Figure 27
"3-input, 3-output cascade functions."
Sasao, 216.

Figure 28
"Cross-stitch/half-cross-stitch."

Figure 29
"All possible combinations and permutations of first five "gestures" in half-cross-stitch."

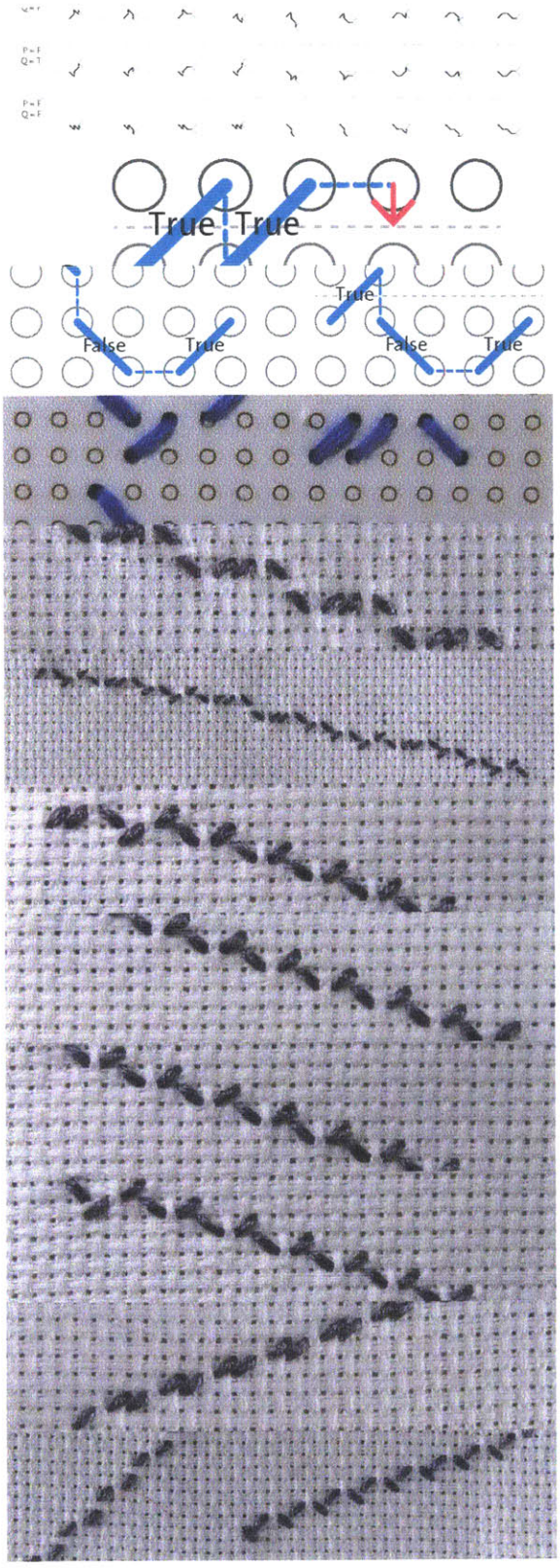


Figure 30
"Combinations and permutations that follow convention."

Figure 31
"The NAND Stitch heuristic algorithm."

Figure 32
"The NAND Stitch with all possible (and conventional) inputs."

Figure 33
"The NAND Stitch with all possible (and conventional) inputs in actuality."

Figure 34
"An NAND Stitch computation started with an input pair: TRUE, FALSE, and continued with TRUE Stitches."

Figure 35
"NAND Stitch chain with random string of inputs."

Figure 36
"An NAND Stitch computation started with an input pair: TRUE, TRUE, and continued with the opposite of whatever the last value stitch was."

Figure 37
"An NAND Stitch computation started with an input pair: TRUE, FALSE, and continued with the opposite of whatever the last value stitch was."

Figure 38
"An NAND Stitch computation started with an input pair: FLASE, FALSE, and continued with the opposite of whatever the last value stitch was."

Figure 39
"An NAND Stitch computation started with an input pair: FALSE, TRUE, and continued with the opposite of whatever the last value stitch was."

Figure 40
"The two variations of the NAND/AND Stitch predicted by Figure 31. Each depicts a chain of TRUE values. From the top, each chain relates to: d/r - towards (NAND); r/d - towards (NAND); d/r - away (AND); r/d - away (AND) ."

Figure 41
"The two variations of the OR/NOR Stitches predicted by Figure 31. Each depicts a chain of TRUE values. From the top left and moving counter-clockwise, each chain relates to: u/r - towards (OR); r/u - towards (OR); u/r - away (NOR); r/u - away (NOR) ."