1 Introduction

Many literary critics have claimed algorithms drive conceptual writing. In a summer 2012 essay, Johanna Drucker defined conceptual writing as "an intellectual product [. . .] indicative of our thought-forms" with "the aesthetic sensibility of rule-based work" that "finds expression in the computational games that parallel algorithmic processing." Several reviewers of Goldsmith's *Uncreative Writing* have discussed his theories as reducing all of literature to algorithms, even though the word "algorithm" appears nowhere in the book. However, it does appear in an April 2010 post to Harriet where Goldsmith describes a new writing emerging, "one of algorithmic rationality and machine worship."

This led Kristen to approach a computer scientist about the question of algorithms. Do conceptual pieces really seem to "parallel algorithmic processing"? We decided rather than accepting algorithms as a metaphor, we would take it literally, and try to describe some conceptual works using the rhetoric of computer science. This post summarizes a number of discussions examining the similarity between the processes in conceptual writing and algorithms in computer science.

2 Definitions

In computer science, an algorithm refers to a set of instructions for performing some task. There are algorithms for sorting numbers, morphologically analyzing words, and processing images. Algorithms can also be compared. For example, an algorithm for searching the web will include steps which involve morphological analysis (when representing documents and queries) and sorting numbers (when ranking documents). Much of computer science as an academic discipline focuses on developing new algorithms, either to improve the solution to an existing task or because the task is altogether new.

An algorithm is primarily evaluated by its ability to perform the task. A sorting algorithm that orders items correctly is better than another algorithm that includes errors; a translation algorithm that accurately translates a document is superior to one which makes errors. Sometimes the metrics are straightforward as in sorting; other times they are more nuanced as in document translation. In addition, algorithms are often evaluated by auxiliary metrics such as speed or memory usage. There are many metrics and rarely do algorithms dominate each other across all metrics.

Throughout our analysis, we will use notation found in introductory computer science texts. We refer to the individual executing the algorithm as the *operator* of algorithm. Algorithmic descriptions will often use the control concepts such as iteration (e.g. **for** loops) and conditional execution (e.g. **if** statements). An algorithm will also include references to supporting data which will be interpreted as passing control to another algorithm. When a function name is prefixed by OPERATOR, control is passed to the human operator(s) to perform the subtask. All other subtasks have well-known implementations. Corpora, be they input, output, or auxiliary texts, are represented as sequences of sentences (i.e. $\mathcal{C} = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$). Corpora may be iterated over (e.g. "**for** $s \in \mathcal{C}$ . . . ") or concatenated (e.g. "$\mathcal{C} \cap \{s\}$"). Our descriptions include comments, indicated by the symbol "▷".

3 Algorithms in Conceptual Writing

In order to support the perspective of conceptual writing as algorithmic, we will describe several examples of conceptual writing using algorithmic notation. All of the works in this section demonstrate algorithmic processes which translate an input corpus by a series of instructions into an output text.

Although these descriptions focus on the mechanical aspects of the works, all algorithms require some operator direction (beyond designing the algorithm itself). Operator intervention comes in two flavors. First, algorithms sometimes require an operator to perform some subtask. As a result, these subtasks are subject to the particular operator's behavior, by explicitly or implicitly massaging the output. Adding in natural operator error, the resulting realizations of a conceptual work may be different from operator to operator. Second, algorithms cannot be realized without the operator defining the input parameters and corpora. Despite our perspective of algorithms as agnostic to input, an operator's choice of parameters—be it characterized as explicit decisions or environmental influences—is fundamental to the realization and—arguably—fundamental to understanding the realization.

3.1 The Weather

In *The Weather*, Goldsmith transcribed weather reports broadcast on 1010 WINS, a New York area news station. *The Weather* exhibits no apparent transformation of the input text beyond segmentation into chapters; the operator merely copies input data to an output device. Goldsmith introduces the work as part of a theme of "transcription, retyping, copying; moving language from one place to another as a valid writing practice".

Algorithmically, we can represent *The Weather* as a sequence of transcription actions performed by the operator observing the input.

```
GoldsmithProcess_0(C)
▷ initialize the output
for s ∈ C
    ▷ iterate over sentences in C
    do
    ▷ pass control to operator
    ▷ concatenate output
    ▷ return the output
```

All variables exogenous to the algorithm are contained in the OPERATORTRANSCRIBE procedure. The algorithm underlying *The Weather* makes no assumption about the input text or the execution of OPERATORTRANSCRIBE; those are under the jurisdiction of the operator.

The specific realization of GOLDSMITHPROCESS$_0$ in *The Weather* depends not only by the curatorial choice of using 1010 WINS weather reports as $\mathcal{C}$ but also the execution of OPERATORTRANSCRIBE. Because this function is subject to operator inconsistency, a new realization of GOLDSMITHPROCESS$_0$, even if it matched the same input, would undoubtedly be different from the published version. With this caveat, we can present this specific realization of GOLDSMITHPROCESS$_0$ as

$$\text{GoldsmithProcess}_0(\mathbf{W})$$

where $\mathbf{W}$ represents the input corpus of weather reports.

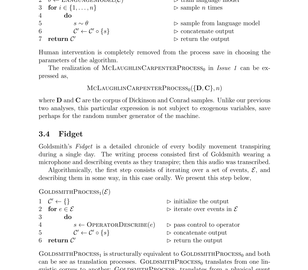
3.2 Apostrophe Engine

Darren Wershler and Bill Kennedy's *Apostrophe Engine*—subsequently published by ECW Press as *Apostrophe: The Book*—is a website that allows the reader to click on any line of the existing poem, spawning processes that would take the line as a search query, filter the resulting pages for phrases beginning "You are" (hence, apostrophe), code-strip them, and present them as a new and enduring section of the poem, with each line published so that the process could be repeated. Insofar as the readers influence the realization of the algorithm, we consider them operators of the algorithm.

Algorithmically, *Apostrophe Engine* takes as input a seed poem, an auxiliary corpus, and a pattern to search for in the results. The core algorithm responds to a stream of operator input (i.e. clicks on sentences).

```
WershlerKennedyProcess_0(C, P, p)
▷ initialize the output
while true
    ▷ infinite loop
    do
    ▷ pass control to operator
    ▷ run query
    ▷ sentences matching p
    ▷ concatenate output
```

As with GOLDSMITHPROCESS$_0$, this algorithm is agnostic to operator decisions, be they in corpus selection or sentence selection (line 4). In fact, any two realizations of WERSHLERKENNEDYPROCESS$_0$ will almost certainly not be equivalent. Primarily, this results from the OPERATORINPUT subtask which, more so than GOLDSMITHPROCESS$_0$, exhibits nondeterminism because the reader selects one of the $|\mathcal{C}|$ sentences at each iteration. Accordingly, realizations of the work diverge very quickly with the number of iterations. Even if two realizations are equivalent at some point in time, they will almost certainly diverge because of the infinite loop (line 2). We present the distribution over similarities of two independent realizations of WERSHLERKENNEDYPROCESS$_0$ in the following figure.



This model assumes that the users in line 2 choose randomly amongst the existing links on the page. Because content is more similar at the beginning of the realization (10 clicks), there is still some overlap. As the number of clicks from users increases, this overlap decays significantly. This divergence will accelerate if we make more realistic assumptions about distinct reader populations and dynamic auxiliary corpora.

The specific realization of WERSHLERKENNEDYPROCESS$_0$ in *Apostrophe Engine* can be represented,

$$\text{WershlerKennedyProcess}_0(\mathbf{A}, \mathbf{W}, \text{"you are"})$$

where $\mathbf{A}$ is the original apostrophe poem and $\mathbf{W}$ is the web. Because the web is dynamic, the output will reflect of the state of the web during the period of realization. Variability across realizations also results from the incorporation of reader input, emphasizing the operator effect discussed in *The Weather*.

3.3 Issue 1

The briefly-infamous *Issue 1*, edited by Stephen McLaughlin and Jim Carpenter, was a pseudo-anthology that randomly assigned poets' names to texts generated by Carpenter's "Erica T. Carter", an implementation of a language modeling algorithm commonly used in disciplines such as speech recognition, machine translation, and information retrieval. These models of sequence data encode statistical regularities in cooccurrence patterns. One important side-effect of language modeling is the ability to generate novel sentences consistent with the observed data. For *Issue 1*, the authors use texts from Emily Dickinson and Joseph Conrad to train the model and generated text statistically similar to these input corpora.

McLaughlinCarpenterProcess$_0(\mathcal{C}, n)$

```
1  C ← {}                                  ▷ initialize the output
2  θ ← LanguageModel(C)                    ▷ train language model
3  for i ∈ {1, . . . , n}                  ▷ sample n times
4      do
5          s ∼ θ                           ▷ sample from language model
6          C ← C ∩ {s}                     ▷ concatenate output
7  return C                                ▷ return the output
```

Human intervention is completely removed from the process save in choosing the parameters of the algorithm.

The realization of MCLAUGHLINCARPENTERPROCESS$_0$ in *Issue 1* can be expressed as,

$$\text{McLaughlinCarpenterProcess}_0(\mathbf{D}, |\mathbf{D}|, n)$$

where $\mathbf{D}$ and $\mathbf{C}$ are the corpus of Dickinson and Conrad samples. Unlike our previous two analyses, this particular expression is not subject to exogenous variables, save perhaps for the random number generator of the machine.

3.4 Fidget

Goldsmith's *Fidget* is a detailed chronicle of every bodily movement transpiring during a single day. The writing process consisted first of Goldsmith wearing a microphone and describing events as they transpire; then this audio was transcribed.

Algorithmically, the first step consisted of recording ones a set of events, $\mathcal{E}$, and describing them in some way, in this case orally. We present this step below.

```
GoldsmithProcess_1(E)
1  C ← {}                                  ▷ initialize the output
2  for e ∈ E                               ▷ iterate over events in E
3      do
4          s ← OperatorDescribe(e)         ▷ pass control to operator
5          C ← C ∩ {s}                     ▷ concatenate output
6  return C                                ▷ return the output
```

GOLDSMITHPROCESS$_1$ is structurally equivalent to GOLDSMITHPROCESS$_0$ and both can be see as translation processes. GOLDSMITHPROCESS$_1$ translates from one linguistic corpus to another; GOLDSMITHPROCESS$_0$ translates from a physical event corpus to a linguistic corpus. The second step, the transcription of the audio, is an instance of GOLDSMITHPROCESS$_0$. For that reason, the finished algorithmic description can be expressed as,

$$\text{GoldsmithProcess}_0(\text{GoldsmithProcess}_1(\mathcal{E}))$$

which should be read as "use the output from a realization of GOLDSMITHPROCESS$_1$ as the input to GOLDSMITHPROCESS$_0$".

The realization of this algorithm in *Fidget* can be expressed as,

$$\text{GoldsmithProcess}_0(\text{GoldsmithProcess}_1(\text{OperatorFidget}()))$$

where OPERATORFIDGET represents the operator's movement through physical space. The GOLDSMITHPROCESS$_1$ process is subject to variability for the same reasons as GOLDSMITHPROCESS$_0$. Moreover, the physiological and physical environment in which OPERATORFIDGET occurs also introduce variability in output.

3.5 Pad

Zuffanski's *Pad* catalogs the author's success or failure in lifting of each of the items in his apartment with his penis.

Pad can be seen as a version of GOLDSMITHPROCESS$_1$. Whereas *Fidget* uses relatively everyday processes to generate $\mathcal{E}$, the algorithm in *Pad* constructs $\mathcal{E}$ by mechanically iterating over a set of items and executing a task. We parameterize the underlying algorithm according to the set of items, $\mathcal{I}$, and the task to be performed, f.

ZuffanskiProcess$_0(f, \mathcal{I})$

```
1  E ← {}                                  ▷ initialize the output
2  for i ∈ I                               ▷ iterate over items in I
3      do
4          e ← OperatorPerform(f, i)       ▷ perform the task f on item i
5          E ← E ∩ {e}                     ▷ concatenate output
6  return E                                ▷ return the output
```

Zuffanski implements OPERATORDESCRIBE as a constrained procedure (i.e. 'My dick [can,cannot] lift [object].').

The realization of the text of *Pad* can be expressed as,

$$\text{GoldsmithProcess}_1(\text{ZuffanskiProcess}_0(\text{OperatorLift}, \mathbf{O}))$$

where OPERATORLIFT represents the task to be performed and $\mathbf{O}$ is the set of objects in the apartment. Like OPERATORFIDGET, the physiological environment affects OPERATORLIFT. On the other hand, the operator's physical environment defines $\mathbf{O}$. The output reflects the moment of realization.

3.6 Ad Pedem Litterae

In *Ad Pedem Litterae*, Giffin describes his own retyping of Goldsmith's *Fidget*, with the implication that potential subsequent iterations of *Ad Pedem Litterae* apply the same function to the output of the previous iteration. So each subsequent iteration would be a description of the retyping of the description of the retyping of the description of the retyping... of *Fidget* (the original *Fidget* being iteration 0). In this case, the algorithm is composed completely of the algorithms studied so far.

GiffynProcess$_0(\mathbf{F}, n)$

```
1  C ← C                                   ▷ initialize the output
2  for i ∈ {1, . . . , n}                  ▷ process n times
3      do
4          C ← ZuffanskiProcess_0(GoldsmithProcess_1(C'))
                                            ▷ perform process
5          C ← GoldsmithProcess_1(C)       ▷ describe process
6  return C                                ▷ return the output
```

where n is the iteration number being generated.

The realization of *Ad Pedem Litterae* iteration n can be expressed as,

$$\text{GiffynProcess}_0(\mathbf{F}, n)$$

where $\mathbf{F}$ is Goldsmith's realization of *Fidget*. Although GIFFYNPROCESS$_0$ inherits the variability in the supporting algorithms, Giffin realized *Ad Pedem Litterae* by using a computer program to automatically replace letters with descriptions of typing those letters. Giffin had to stop the process after the first iteration because of computational constraints. We can see why this happens by noticing that the output size is exponential in the number of iterations, n. We present a plot of the output size as a function of iteration number below,

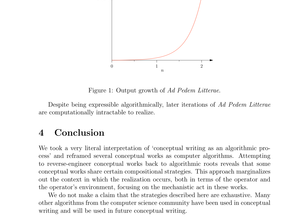


Figure 1: Output growth of *Ad Pedem Litterae*.

Despite being expressible algorithmically, later iterations of *Ad Pedem Litterae* are computationally intractable to realize.

4 Conclusion

We took a very literal interpretation of 'conceptual writing as an algorithmic process' and reframed several conceptual works as computer algorithms. Attempting to reverse-engineer conceptual works back to algorithmic roots reveals that some conceptual works share certain compositional strategies. This approach marginalizes out the context in which the realization occurs, both in terms of the operator and the operator's environment, focusing on the mechanistic act in these works.

We do not make a claim that the strategies described here are exhaustive. Many other algorithms from the computer science community have been used in conceptual writing and will be used in future conceptual writing.

5 Exercise

Use the approach from Section 3 to write the algorithm for the following piece,

http://inhumanscale.bandcamp.com/track/ha003-sibling-10