humanities

MDPI

Article

IBM Poetry: Exploring Restriction in Computer Poems

Christopher T. Funkhouser

Department of Humanities, New Jersey Institute of Technology, University Heights Newark, NJ 07102, USA; christopher.t.funkhouser@njit.edu; Tel.: +1-973-596-6335

Academic Editors: Burt Kimmelman and Philip Andrew Klobucar
Received: 17 October 2016; Accepted: 28 February 2017; Published: 8 March 2017

Abstract: In the 1960s, many years prior to the advent of personal computers and mainstream cultural accessibility to them, Emmett Williams devised a method that he felt reflected the expressive potential of algorithmic processes within a printed page's confines. Williams' "IBM" method serves as a "muse's assistant," in which a user-contrived vocabulary is employed to construct poems in which letters of words in one line are used to create subsequent lines. This article introduces the imposed conditions of Williams' invention, comparing and placing them within a range of digital writings that appear during subsequent decades.

Keywords: computer poetry; algorithmic composition; permutational writing; stochastic text

What is this confused-looking, four-armed robot with a computer on its head, manically pushing buttons (Figure 1)? The image, titled "Cybernetics," accompanies computer poetry's first reference in North American mainstream media, in an article titled "The Pocketa, Pocketa School" appearing in *Time* magazine in 1962. In certain ways, digital poets have come to embody the histrionic, exaggerated creature depicted, and what catches my eye when returning to this short article is the image's caption, which reads, "A.B. Computer? Unfree verse" (A.B. stands for "Auto Beatnik," the program reviewed in *Time*) ([1], p. 99). Contemplating restriction in computer poems, I am particularly intrigued by encountering the phrase "unfree verse," although in context the article is commenting at the high price of computing in the early 1960s.[1] While the tone of the article is curious, and concludes by surmising, "there is a chance of a whole new school of poetry growing up," it essentially makes fun of the process, considering the matter with only *faux* seriousness. It suggests, interestingly, "the machine needs help"—a point I happen to agree with in the context of writing algorithmic poems.

The article's contents directly intersect with my thinking on the topic, in arguing that "by drastically cutting down its choice of words—so that the incidence of a subject word reappearing is greatly increased—engineers can make the machine seem to keep to one topic" ([1], p. 99). Therefore, in ostensibly the very first piece of criticism on the subject, creating a computer poem was seen as something that involves limitation in order to succeed (or, it could perhaps be said, even to exist). This idea, of course, flies in the face of the viewpoint that celebrates the endless possibilities of electronic text, sometimes heard in discussions on the topic.

In light of the above, my present discussion will extend what I have written in the past.[2] Below I discuss works I have covered previously and introduce new historical materials and areas of inquiry. Hence this essay is a remix of sorts, containing information pertinent to the topic at hand. I am a

[1] The article claims the auto-beat computer costs $100,000, thus "the output will certainly not be free verse" ([1], p. 99).
[2] My book [2] *Prehistoric Digital Poetry: An Archaeology of Forms 1959–1995* (Alabama, 2007) discusses "Auto Beatnik" and dozens of other examples of early computer poems.

critical proponent of digital poetry and celebrate expressive possibilities that can be reached through a variety of mechanisms, but I am also closely aware of the fact that, in many cases, if not most, the digital poem exists within tight boundaries, even if they are inventive. This has become obvious to me through both research and creative practice.

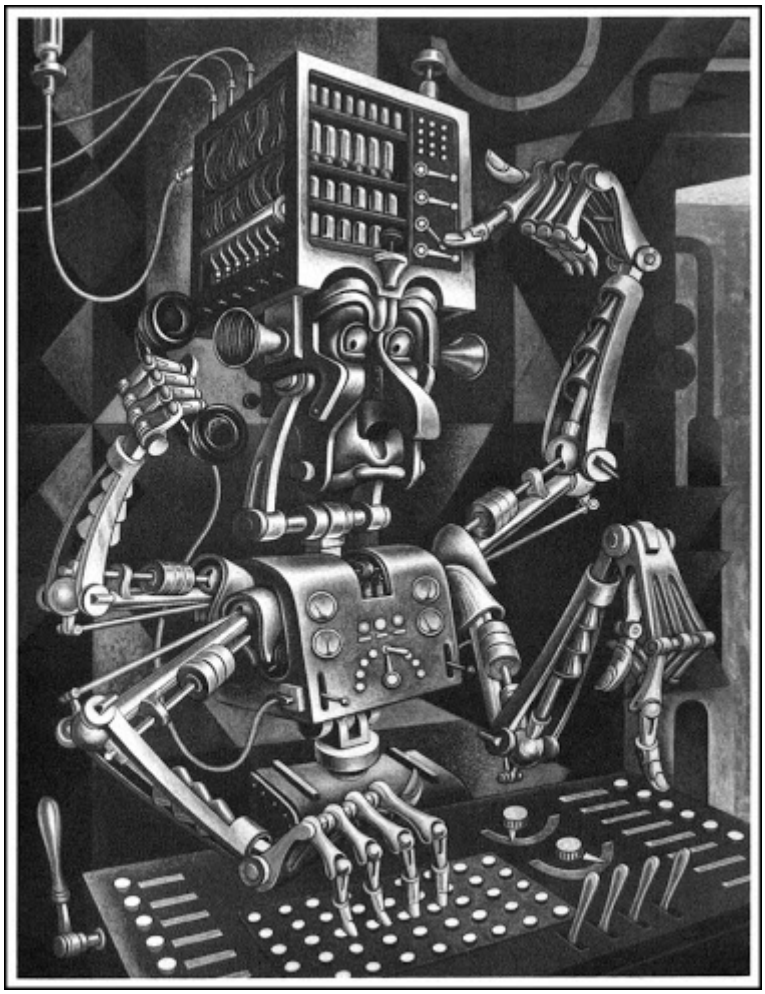

Figure 1. Boris Artzybasheff. "The Pocketa, Pocketa School". *Time*, 25 May 1962 ([1], p. 99).

With that in mind, I begin by introducing one of the very first computer poems created by an American, Emmett Williams's "IBM," which is documented and versioned in his book *A Valentine for Noël* [3]. In 1966, Williams programs a poem he had originally formulated in 1956; the process, as described in *A Valentine for Noël*, involves randomly choosing twenty-six words (which he does by chance, although he admits he may have cheated) and then associating each of them with a letter of the alphabet to create "an alphabet of words" ([3], p. 3). A three-letter title is chosen, and the first line of the poem is determined by substituting words for letters in the title. Letters of words in one line are then used to make subsequent lines.

To demonstrate and practice the mechanics involved in this poem is the best way to become familiar with the types of conventions imposed on machinated works. To that end, I have prepared many "IBM Poems" and have created an analog template for "IBM" processing for students to use.[3]

[3] See [4] for template.

The vocabulary in Williams's original version of the poem produces (beginning with "IBM" as input): "red up going" as its title, followed by the lines "perilous like sex/yes hotdogs/evil jesus red black evil" ([3], p. 5). These ten words and forty-six letters lead to a ten-line, forty-six word poem; the volume of the poem expands exponentially. Williams, however, was not interested in this possibility beyond processing the letters of each of the words in a second set of lines one more time—probably because configuration of the lines becomes quickly redundant as a result of the small vocabulary. He writes of the computer as "the muse's assistant," and that the poem as such amounts to an "eternal project," but he considered it "no great accomplishment" ([3], p. 4). In the rendition of "IBM" shown in *A Valentine for Noël*, Williams enhances the process by imparting a "cylindrical" effect by shifting the vocabulary attached to the letter after the third process of substitution is made (i.e., at that juncture the initial "a" word becomes the "b" word, "b" becomes "c," and so on) to create new poems. In the version of "IBM" that appears in the book, each letter is matched with each word in the vocabulary one time, so twenty-six distinct pairs of poems appear, each stemming from "IBM" as input. A visual aspect was added when Williams used a Diatype machine to increase the size of the word each time it is repeated, as seen in Figure 2, a 1973 example of a second generation poem originating from the phrase "fear hotdogs money."

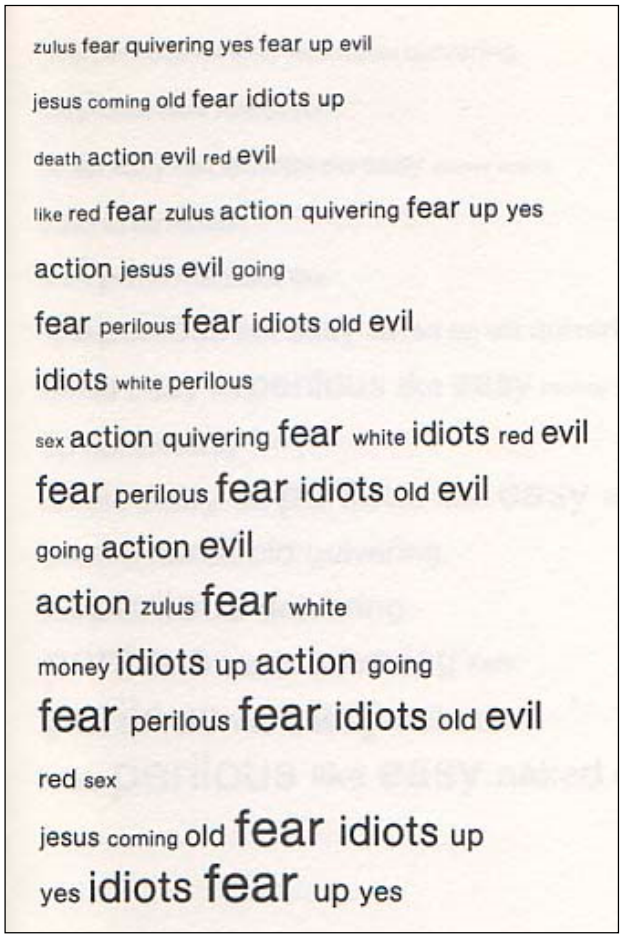

Figure 2. Emmett Williams. *A Valentine for Noël*. ([3], p. 33).

In these compositions, the repetition of specific words leads to repetition of lines, which would not be as interesting were it not for the visual component. In contrast to Brion Gysin's earlier permutations, discussed below, Williams's combinatoric texts feature self-contained generative dimensions at the level of the letter. Words are presented not to convey syntactically correct meaning, but are the result of the combination of letters found in a preceding iteration of the contrived structure, and are reused as

many times as the letter initially appears. As they are repeated and increase in size, a verbal and visual amplification is presented. The word thus contains more weight and becomes, perhaps, thematic (as do "fear" and "idiots" in Figure 2). This process increases the variables, and, using a limited database (twenty-six listed words in "IBM"), a vast number of different poems can be made from a small amount of input text. Williams crafts the poem by formulizing the work and adding a visual framework, making it something other than an oversized permutation fraught with repetition. His use of a specific, pre-meditated equation(s), as did artists and writers involved with Oulipo, adds qualities to the work that would not be present otherwise.[4]

Experimenting with the equation established in the program, I prepared an IBM poem for a performance at the National Science Foundation 2008 meeting on Codework. Here are the first few lines ([indicates continuation of the previous line):

space aspire dispense (GRU)

spatial perspective aspects corresponds especially
aspects spatial perspective spoke aspire especially
despite spoke spatial perspective especially specifics spatial especially
spatial perspective especially corresponds spoke aspects Brazil

perspective especially aspire spatial perspective especially corresponds

[spontaneous spoke atmosphere especially

aspects spatial perspective especially corresponds spontaneous spatial

corresponds news aspire aspire especially spatial perspective news

[specifics despite spatial

especially spatial perspective especially corresponds spoke

[aspects Brazil Brazil position

aspects spatial perspective especially corresponds

[spontaneous spatial

spatial perspective especially corresponds spoke aspects

[Brazil

If you experiment with Williams's scheme, you may discover the benefits of using words with fewer letters. Sixty-five percent of the words in Williams's original contain five letters or less and the outcome of such a restriction is clear. As a result of the size of the words in my vocabulary, my poem contains nearly twice as many lines as Williams's first example, many of which seem might seem

[4] The Oulipo group, founded in France in 1960, advanced various forms of non-computerized procedural poetry and writing that employed arithmetic and other programmatic constraints.

redundant—two lines appear three times, and one four times, taking up nearly half of the space of the poem. If you want a bigger poem, you will learn to use the longest words you can find (and discover that the size of the lines, if you want them to be legible, might quickly go beyond the boundaries of a typical printed page).

You may also end up—depending on what kind of poet you are—wanting to strategically manipulate the placement of words in the dictionary, and might also consciously select words which minimally repeat letters. If you want to use all of the words in the vocabulary, you will have to incorporate every letter in the alphabet in the words you choose. While I do not have much trouble with frequent repetition—which is unquestionably a characteristic of computer poetry and which can certainly serve a purpose—using smaller words that, in all, contain as many letters, with as few repetitions of each letter as possible in one's "alphabet of words," will render a more manageable poem suited for someone looking for more verbal diversification. If a letter appears more than once in the title (e.g., "All"), much repetition would follow.

My experiments exhibit few of these approaches, and at first—due to inexperience and a type of thoughtlessness—I did not realize the prominence the word "spatial" would come to have in the poem when I put it in the "s" slot. Most of the words in the vocabulary of the poem, whose name is taken from the initials of the Guarulhos Airport in São Paulo, come from the words of a poem "SP," derived from an interview I did with Jorge Luiz Antonio in 2004—all of which contain the letter combination "sp," which is why "spatial" and "perspective" are so prevalent (many words are associated with the letter they begin with in the database/vocabulary, as in this example). The main visual feature of the Codework performance is a video I shot on the streets of São Paulo, which is accompanied by a soundtrack recorded in Moema, São Paulo.

The only slightly illegitimate word (given my process) inserted into the poem is Brazil—which appears in the interview but not in the poem derived from it. I wanted to give the poem a place. I had some geographical intent when forming the vocabulary but did not know what would happen. As it turns out, despite the fact that the length of the poem's lines exceeds the boundaries of regular-sized sheets of paper, I am actually quite pleased by how the lyric seems to represent more than a single style, or voice, of the twentieth-century. On one hand, it emulates a style akin to Gertrude Stein's work, with cubist-esque repetition of themes and phrases; on the other, words create a visual texture that shares some commonality with concretist aesthetics.

A look at some other earlier works are also revealing in terms of identifying the types of restrictions computer poems exist within, and what authors are able to accomplish given the imposed conditions.

Gysin's "I am that I am" is a cyclical, randomized representation of the three words contained in that phrase. Gysin's interest in the form stems not from technological or mathematical interests but from visual ones. In Richard Kostelanetz's anthology *Text-Sound Texts*, Gysin writes, "The whole idea of the permutations came to me visually on seeing the so-called, Divine Tautology, in print. It looked wrong, to me, non-symmetrical. The biggest word, That, belonged in the middle but all I had to do was switch the last two words and It asked a question: "I Am That, Am I?" The rest followed" ([5], p. 373). This work imposes a pre-established pattern on the words in a phrase, so they appear in different orders until all possibilities have been exhausted. Thus, a poem made with a three-word phrase will be six lines long ($3 \times 2 \times 1$); a poem that begins with a five-line phrase, such as "I am that I am" will be one hundred twenty lines long ($5 \times 4 \times 3 \times 2 \times 1$). The availability of computer technology automated the process of randomizing these permutations. The anthology, Brion Gysin: *Tuning in to the Multimedia Age*, shows four examples of computer-generated permutation poems, programmed with a random sequence generator on a Honeywell computer (using punch cards) by Cambridge University mathematics student Ian Somerville in 1960. As shown in Figure 3, output appears in block formation.[5]

[5] The programming details are not available; alternate versions of the poem, in which the words appear with a different sort of arrangement, are included in Williams's *An Anthology of Concrete Poetry* (1967) [6] and in Kostelanetz's *Text-Sound Texts* (1980) [5].

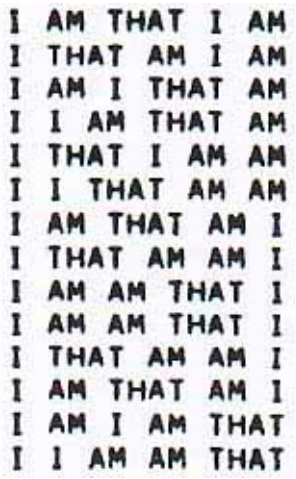

Figure 3. Brion Gysin. Permutation poem 1960 ([7], p. 93).

In "Cut-Ups Self-Explained" (1964) Gysin declares, "The permuted poems set the words spinning off on their own; echoing out as the words of a potent phrase are permuted into an expanding ripple of meanings which they did not seems to be capable of when they were struck and then stuck into that phrase" ([7], p. 154). Gysin's creations resonate with the methods of a "proteus" poem, and can also be seen as an adaptation or transformation of the traditional renga poem. However, instead of keeping an entire line intact, the poet creates a poem from one line, in which the words are internally cycled in a random pattern. In Gysin's work, the process is repeated over and over until every word has appeared in every possible position in the line of the poem. I know of no emulator for Gysin's work, although wonderful audio recordings of the "I am that I am" (1960) and "Junk is no good baby" (1962) permutations are available at Ubuweb [8].

Two real time permutation poems (in Portuguese) by Pedro Barbosa, "Porto" and "Aveiro" (1977), originally made in the 1970s (and appearing in the French hypermedia journal *Alire* in the early 90s), are available within the Syntext-W program online.[6]

In these works, both of which focus on cities in Portugal, strict permutation with added elements are found; the output appears as a block of text of capitalized letters (and as such has a strong visual quality). In "Porto" (Figure 4), the inclusion of four prepositions to accompany the four subjects enables grammatical variation in the output and increases the number of possible outcomes from 24 ($4 \times 3 \times 2 \times 1$) to 40,320. In "Aveiro," the verbal configurations "with" and "without" are added as a way to diversify and vary the way statements are formed around the city's river ("ria") and its water ("água"). Barbosa's programs do enable the words to spin off on their own and are remarkable because endless different phrases are built that transmit different dimensions of the same sentiment. In each of the lines of "Porto," a sense of the passage of time, as absorbed by and reflected in the rock formation that supports the city, is apparent. By extension, other cultural aspects of the city and its people may be read into the lines, like how the people of Porto possess a strong sense of their past, or even how a longing for the past ("the stone of nostalgia") is engrained in the populous. The reader confronts impossible circumstances ("the nostalgia of the stone") made possible only through creative reflex, some of which are challenging to interpret or envision ("in the stone the history of the nostalgia of granite"). Yet out of such nonsense we are able to detect poetic logic. Deeper meaning can be evoked even when odd statements are encountered, especially when the litany is absorbed as a whole and not just as individual lines.

---

[6] See [9]. This version of the work is prepared with Java, which may prevent viewing in some browsers.

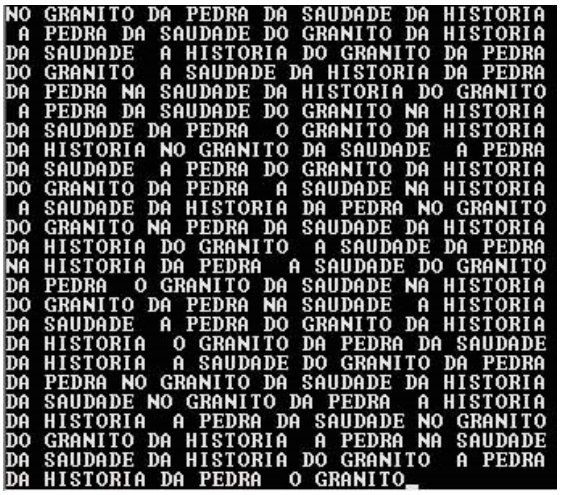

Figure 4. Pedro Barbosa. Porto, 1977.

If we now go back and look at the earliest work in computer poetry, Theo Lutz's "stochastic text" experiments in 1959, we see a sharp contrast in that there is no apparent interest in visuality. The output resembles a generic teletype document (Figure 5):

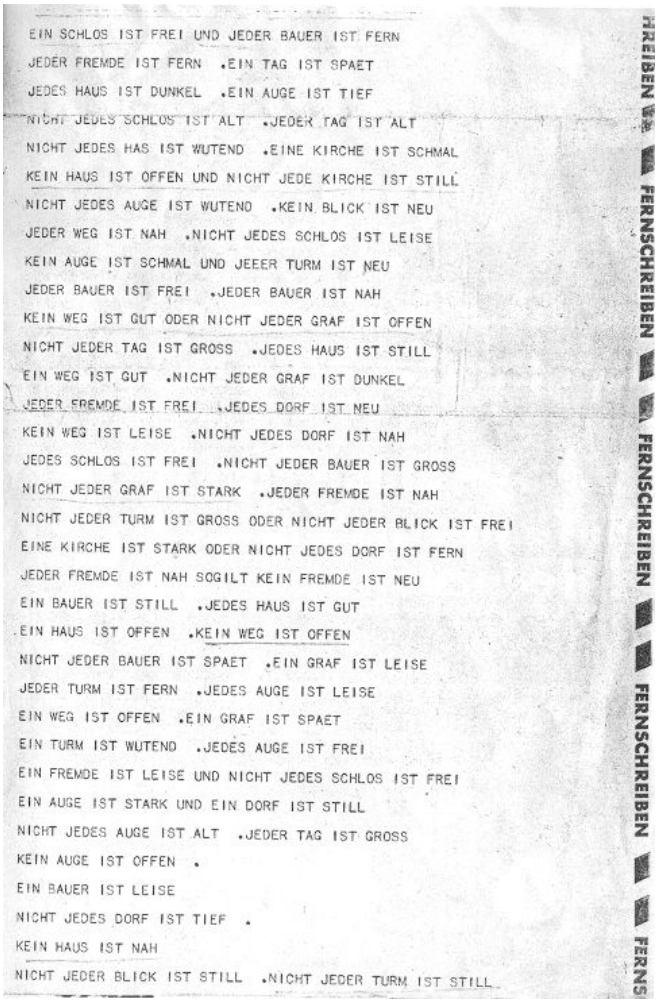

Figure 5. Theo Lutz, Stochastic text.

The impetus for Lutz was mathematics: he made "stochastic" (i.e., random variation) poems written on a program-controlled Zuse Z 22 computer. At the time, he was a student of Max Bense, who suggested using a random number generator to accidentally determine texts.[7] Examples of this work, which applies tools developed for mathematics and calculation (i.e., logical structures) to process language, along with descriptions of its attributes, were published in a 1959 article ("Stochastic Text") in Bense's journal *Augenblick*.[8]

Lutz made a database of sixteen subjects and sixteen titles from Franz Kafka's novel *The Castle*. Lutz's program randomly generated a sequence of numbers, pulled up each of the subjects/titles, and connected them using logical verbal constants (such as conjunction) in order to create plain syntax.[9] Here is a translation from one of Lutz's original "Stochastic Texts":

> Not every look is near. No village is late.
> A Castle is free and every farmer is far.
> Every stranger is far. A day is late.
> Every house is dark. An eye is deep.
> Not every castle is old. Every day is old.
> Not every guest is angry. A church is narrow.
> No house is open and not every church is silent.
> Not every eye is angry. No look is new. (MacCormack)

In this example, we can see patterns and repetitions of words, along with discursive leaps and quirky, unusual semantic connections (e.g., "No village is late").

The words themselves are not complicated, but when they are automatically or randomly arranged into syntax via computer programing the transaction imposes a non-rational ordering of subjects and thoughts. The text—here in translation, a further complication—is readable but disjunctive. Readers must connect and interpret abstractions in the poem (not a new phenomenon in reading or writing poetry), and derive meaning from the verbal associations while reading the text in and against its context. As Friedrich Block and Rui Torres observe in their essay, "Poetic Transformations in(to) the Digital," this work propels Bense's theories at the time, which involved "the turn from idealistic subjectivity to rationalism and objectivity of art" and "the turn from mystic creation to statistic innovation" [16].

Lutz's choice to build the first computer poems based on Kafka's book is interesting, and adds a layer of significance to the endeavor. His decision to include so few inputs (sixteen titles, sixteen subjects) may have been technologically driven, but plausibly relates to themes established in *The Castle*; it is possible that Lutz chose Kafka's incomplete novel as a foundation out of respect for poetry, as a way to question the communicative values of routine, machine-modulated verse. While the processes of generating or consuming the poetry do not particularly reflect or require the reader to embody the type of mysterious bureaucracy experienced by the protagonist of the novel, an alienated, barren tone pervades the published output of the program.[10] The best examples of output are successful because the reader, via the poet's condensation and computer processing of the materials, can rediscover the essence of Kafka's story, or somehow experience new perspectives derived from the original text. Lutz's selection of words, combined with his programming method, enables a speculative, self-reflexive, unconventional style of expression; the programming method consists of about fifty commands and could theoretically generate over 4 million different sentences.

---

[7] Three years later Bense published one of the first essays about composing "artificial" poems, "Üeber naturüliche und Kúnstliche Poesie" ("On natural and artificial poetry"), published in *Theorie der Texte* (*Text Theory*), 1962 [10].

[8] The article was published in *Augenblick* 4 (1959) and is republished on the WWW at [11]; see also [12].

[9] Artistic programmers have successfully created online emulations of the work. German artist Johannes Auer has created two websites that emulate the program online: [13,14] and Nick Monfort has created a Python version, [15].

[10] Lutz also published examples of "Stochastic Text," and similar works, in a book he co-wrote with Rolf Lohberg titled *Electronic Brain*. Examples of his "Autopoems" are also found in Abraham Moles' *Art et Ordinateur* (*Art and Computer*) (1971), and in Pedro Barbosa's *A Ciberliteratura* (1996).

"Stochastic Text" merges sparse, pre-set word lists in controlled and random combinations. The language also contains permutation: the same few words are used over and over, each time the program is run. It is not a permutation of the entirety of Kafka's text; rather, it is a variable, fragmented permutation of the words Lutz chose from the story. Lutz was at the crest of a wave that viewed mathematics, science, and creativity as cooperative disciplines, through which new interrelationships could be forged through computer mechanics.

Many examples exist, and one can easily see the historical role Dada and Oulipo appear to play in the development of this form of expression. I could also continue by elaborating on works by two well-known artists, John Cage and Jackson Mac Low, who turned to computer poetry once it became available to them. Both Cage and Mac Low, at phases in their careers, created randomized poetry with significant graphical elements; they created extremely inventive and profound poems which are curious aberrations of form derived by using and working within established parameters. Both poets used computer programs to facilitate work that they had previously performed manually, and admitted to using a significant degree of systematic editing. Both used input texts that were written by other authors—the original texts are formulaically torn apart and reconfigured—and both also considered their works to be musical compositions in addition to being poems. These works are multimedia conceptions in which the most advanced practitioners of digital poetry using media to extend the parameters of a written work so that it takes on alternative identities (within a new set of parameters). Poetry has always had an association with lyricism, but in these works the words are no longer vehicles for semantic meaning, but instead are literally sound bites (and bytes).

In the end, digital works imposing strict constraint often employ mechanical permutation. Permutation indicates limitation, yet there is not only permutation, but mutation, a fluctuation advanced by the visual attributes in works as I have discussed elsewhere and that are found in the output of text generators and in static and kinetic visual works.

For instance, Jim Andrews's "Snapshot in the Continuing Adventures of I" (Figure 6) inscribes the morphing and mutation of a single word (with the "i" removed) through space. Brian Kim Stefans' masterful "The Dreamlife of Letters" is a calculated, animated manipulation and permutation of words from a piece of writing by Rachel Blau duPlessis. In Stephanie Strickland's *slippingglimpse*, which is interactive and non-linear, a succinct set of textual and videographical material is at the disposal of the viewer, who essentially re-orders the information provided by the poet.

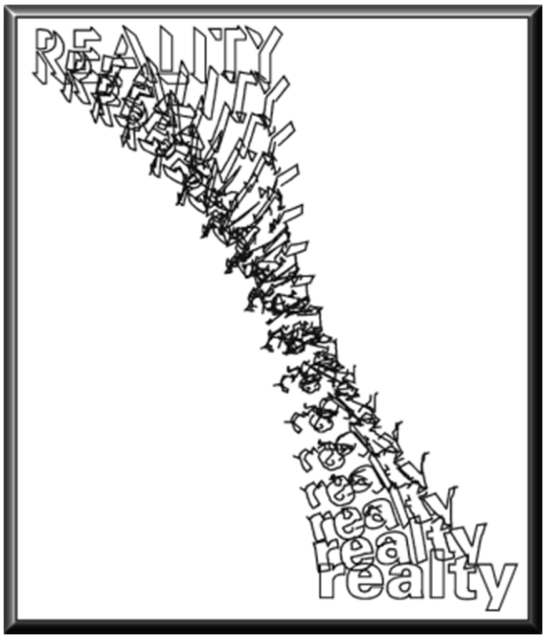

Figure 6. Jim Andrews, "Snapshot in the Continuing Adventures of I" 1990.

Numerous historic and contemporary intermedia experiments reflect this dynamic, which are abundant, and enabled by the technologies used, on the Web. Online tools, such as the Internet Anagram Server, is a marvelous and useful permutation program, which quickly provides all of the anagrams that can be made with a word or phrase, and can be used to identify all of the words that can be created with the letters contained in a phrase [17]. I use this apparatus to create text-movies of various sorts. Such works reflect aesthetic expansion but the retention of restriction through compositional choices imposed, a mediated version of the type of re-patterning practiced by poets for centuries.

There is an imposed "world" on all these poems, in which the authors actually "impose" a variety of filters and gates in the construction of poetry, which gives the work its substance and direction. Confines of the encodings are revealed by the content and output that is viewed. The evolution of software engineering has probably influenced a sense of restriction in digital writing practices, but even some of the best works coded for the Web, such as Jim Carpenter's *Erika* (no longer available) or Charles O. Hartman's *Pyprose* [18] (which uses an entire dictionary to cull textual elements), are unable to fully break away from redundancy and repetition. In the analog world, on the other hand, it is clear that certain code artists, such as Mary Ann Breeze (mez)—who uses her own hybridized 'net speak' language, mezangelle, to convey themes and propel dialogue in her work—have been able to conceive of an expansive, flexible sense of language without writing in computer languages or using sophisticated software programs. In mezangelle, Mez employs text and text format manipulation to give an organized set of letters multiple constructions; using braces and square brackets, as well as other symbol and punctuation keys often found in programming code, she customarily divides and imaginatively re-joins fragments of words.

Working with programs that automate encoding has led authors to create both simple and complicated original approaches, many of which are not meant to be expansive but rather are made for contemplative or educational purposes, or to make social commentary through multimedia, or perhaps just for experiment or folly. Considering these issues of textual confinement alongside aesthetic expansion and formal transformation in poetry, can we, by somehow coding a vast text, possibly go beyond confronting a certain, predetermined set of circumstances in the digital poem? Or is the recognition, understanding, and identification of the confinement itself what establishes the form, allows genre typologies to be built, and formal advancements to be made? I continue to search for digital poetry, or a system of digital poetry, which not only contains self-generated aspects—something that contains a mechanism that allows for the expansion of expressive qualities in any text, or perhaps even from "scratch." To this end, a tool being developed by David Ayre and Andrew Klobucar called the "GTR Language Workbench" may prove itself valuable to writers in the future, similar to the ways in which Photoshop is valuable to computer graphics artists and others at present. By incorporating a wide range of applicable filters within a single program, interactive applications such as the Language Workbench are able to treat any input text with a variety of processes.

To conclude, let me say that, if what Brian Stefans suggests in *Fashionable Noise* is true, then authors of computer poetry will "aspire" to make texts that feature convention because of the works' active and "parasitic" relationship to its primary hosts, publishing and academic institutions ([19], p. 148). Setting aside some obvious projects that prove there are numerous artists who are their own hosts (e.g., VISPO, or Young-Hae Chang Heavy Industries), and are making digital poems for their own creative purposes and reasons, I think there is even more to be added to Stefans' solid observation. One could reasonably argue that many digital and computer poems are self-parasitic. I have written elsewhere about digital poetry as a textually cannibalistic form, and that is true—but sometimes what we see, through various forms of permutation, is the text feeding back into itself, consuming itself to progress. And when it is not busy eating and regurgitating itself, it is a text that is very much about itself, containing technologically driven internal commentaries (direct or indirect) through the text or textual objects. It is clear that digital poetry (and all instances of computer poetry) tend to feed off of something, whether it is historical poetry or other inscribed texts, or itself. I would not qualify this view

as necessarily conservative, or negative, but instead a matter of fact. The self-reflexive nature of works created in the genre's early years may have prevented expansion—although there is certainly artistry within it. Authors have proven that the implementation and combination of good input texts can lead to the creation of inventive electronic poetry. Thoughtful consideration, and embracing limitation, gives rise to form, to the identification of structure, and indeed to the plausibility of a genre of writing that includes multimedia and interactivity.

Conflicts of Interest: The author declares no conflict of interest.

References

1. "The Pocketa, Pocketa School." *TIME*, 25 May 1962, p. 99.
2. Funkhouser, Christopher Thompson. *Prehistoric Digital Poetry: An Archaeology of Forms, 1959–1995*. Tuscaloosa: University of Alabama Press, 2007.
3. Williams, Emmett. *A Valentine for Noël: Four Variations on a Scheme*. Barton: Something Else Press, 1973.
4. Funkhouser, Christopher Thompson. "Template for creating IBM Poetry." Available online: https://web.njit.edu/~funkhous/IBM_TEMPLATE.pdf (accessed on 6 March 2017).
5. Kostelanetz, Richard, ed. *Text-Sound Texts*. New York: William Morrow, 1980.
6. Williams, Emmett, ed. *An Anthology of Concrete Poetry*. New York: Something Else Press, 1967.
7. Kuri, José Férez, ed. *Brion Gysin: Tuning in to the Multimedia Age*. New York: Thames and Hudson, 2003.
8. Ubuweb. Available online: http://www.ubu.com/sound/gysin.html (accessed on 7 March 2017).
9. Barbosa, Pedro. "O motor textual. Livro virtual / infinito." Available online: http://po-ex.net/taxonomia/materialidades/digitais/pedro-barbosa-o-motor-textual-livro-virtual-infinito (accessed on 7 March 2017).
10. Bense, Max. "Üeber naturüliche und Kúnstliche Poesie (On natural and artificial poetry)." In *Theorie der Texte (Text Theory)*. Köln: Kiepenheuer & Witsch, 1962.
11. Lutz, Theo. "Stochastische Texte." Available online: http://www.reinhard-doehl.de/poetscorner/lutz1.htm (accessed on 7 March 2017).
12. Lutz, Theo. "Stochastic Texts." Translated by Helen MacCormack. Available online: http://www.stuttgarter-schule.de/lutz_schule_en.htm (accessed on 6 March 2017).
13. "Theo Lutz (23.7.1932—31.1.2010): Stochastische Texte (1959)." Available online: http://auer.netzliteratur.net/0_lutz/lutz_original.html (accessed on 7 March 2017).
14. Lutz, Theo. "stochastische texte 1959." Available online: http://copernicus.netzliteratur.net/index1.html (accessed on 7 March 2017).
15. Monfort, Nick. "Stochastic Texts (Python version)." Available online: http://nickm.com/memslam/stochastic_texts.html (accessed on 7 March 2017).
16. Block, Friedrich W., and Rui Torres. "Poetic Transformations in(to) the Digital." Available online: http://www.netzliteratur.net/block/poetic_transformations.html (accessed on 6 March 2017).
17. Internet Anagram Server. Available online: http://wordsmith.org/anagram/ (accessed on 7 March 2017).
18. Hartman, Charles O. "Computer program." *Pyprose*. Available online: http://oak.conncoll.edu/cohar/Programs.htm (accessed on 6 March 2017).
19. Stefans, Brian Kim. *Fashionable Noise: On Digital Poetics*. Berkeley: Atelos, 2003.