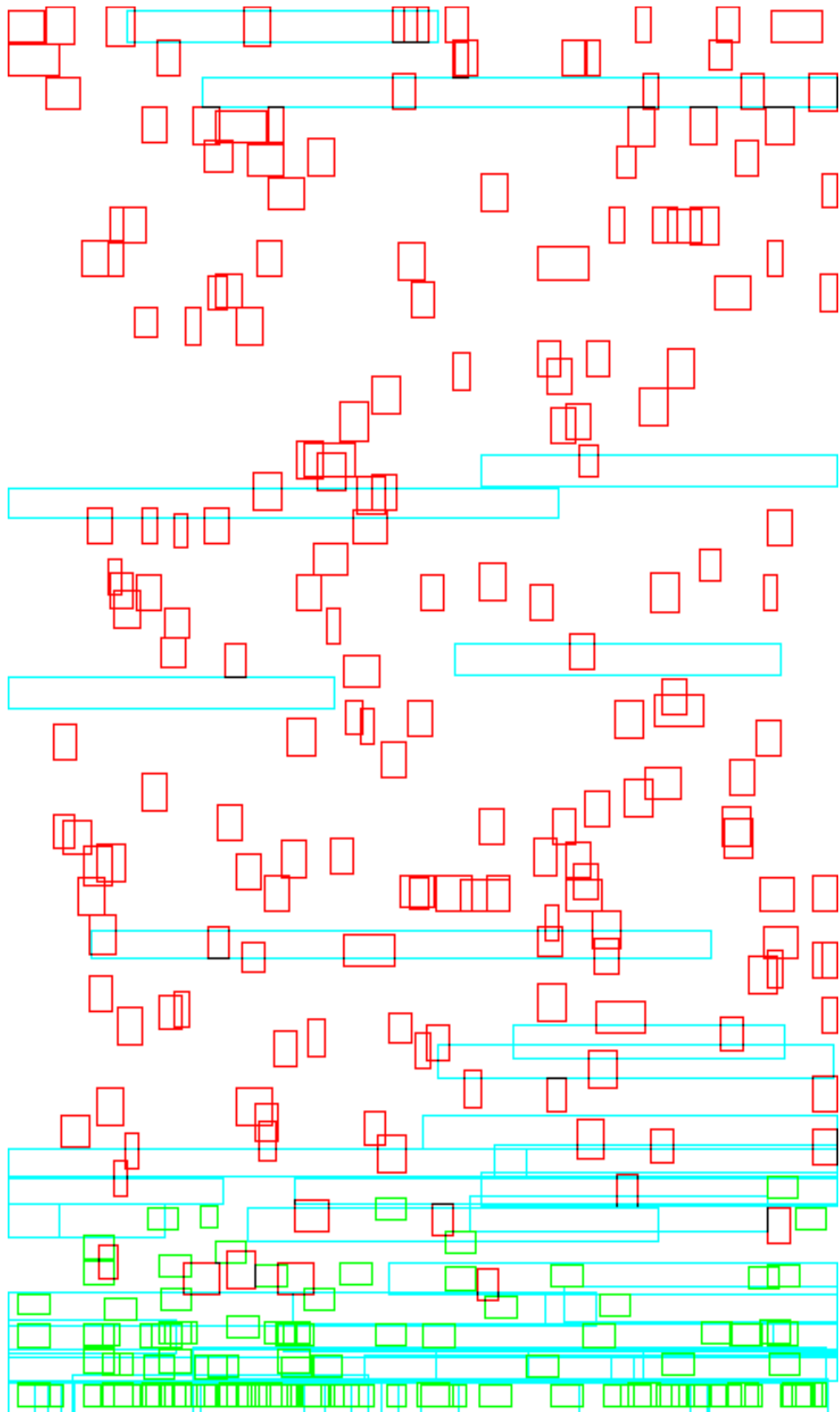


WORDS MADE FLESH

Code, Culture, Imagination

Florian Cramer

Media Design Research
Piet Zwart Institute
institute for postgraduate studies and research
Willem de Kooning Academy Hogeschool Rotterdam



ABSTRACT: Executable code existed centuries before the invention of the computer in magic, Kabbalah, musical composition and experimental poetry. These practices are often neglected as a historical pretext of contemporary software culture and electronic arts. Above all, they link computations to a vast speculative imagination that encompasses art, language, technology, philosophy and religion. These speculations in turn inscribe themselves into the technology. Since even the most simple formalism requires symbols with which it can be expressed, and symbols have cultural connotations, any code is loaded with meaning. This booklet writes a small cultural history of imaginative computation, reconstructing both the obsessive persistence and contradictory mutations of the phantasm that symbols turn physical, and words are made flesh.

Media Design Research
 Piet Zwart Institute
 institute for postgraduate studies and research
 Willem de Kooning Academy Hogeschool Rotterdam
<http://www.pzwart.wdka.hro.nl>

The author wishes to thank Piet Zwart Institute Media Design Research for the fellowship on which this book was written.

Editor: Matthew Fuller, additional corrections: T. Peal

Typeset by Florian Cramer with LaTeX using the *amsbook* document class and the Bitstream Charter typeface.

Front illustration: Permutation table for the pronunciation of God's name, from Abraham Abulafia's *Or HaSeichel (The Light of the Intellect)*, 13th century

©2005 Florian Cramer, Piet Zwart Institute

Permission is granted to copy, distribute and/or modify this document under the terms of any of the following licenses:

- (1) the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version. To view of copy of this license, visit <http://www.gnu.org/copyleft/gpl.html> or send a letter to the Free Software Foundation, Inc., 59 Temple Place—Suite 330, Boston, MA 02111-1307, USA.
- (2) the GNU Free Documentation License as published by the Free Software Foundation; either version 1.2 of the license or any later version. To view of copy of this license, visit <http://www.gnu.org/copyleft/fdl.html> or send a letter to the Free Software Foundation, Inc., 59 Temple Place—Suite 330, Boston, MA 02111-1307, USA.
- (3) the Creative Commons Attribution-ShareAlike License; either version 2.0 of license or any later version. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Your fair use and other rights are not affected by the above.

Contents

Chapter 1. Introduction: In Dark Territory	7
Chapter 2. Computations of Totality	11
Exe.cut[up]able statements	11
Magic and religion	14
Pythagorean harmony as a cosmological code	20
Kabbalah	29
Ramon Llull and Lullism	36
Rhetoric and poetics	41
Combinatory poetry and the occult	47
Computation as a figure of thought	53
Chapter 3. Computation as Fragmentation	57
Gulliver's Travels	58
The Library of Babel	61
Romanticist combinatorics	63
Concrete poetry	65
Max Bense and "information aesthetics"	66
Situationism, Surrealism and psychogeography	70
Markov chains	73
Tristan Tzara and cut-ups	75
John Cage's indeterminism	77
Italo Calvino and machine-generated literature	80
Software as industrialization of art	81
Authorship and subjectivity	83
Pataphysics and Oulipo	88
Abraham M. Moles' computational aesthetics	92
Source code poetry	93
Jodi	95
1337 speech	98
Codework	99
Chapter 4. Automatism and Their Constraints	103
Artificial Intelligence	103
Athanasius Kircher's box	105

John Searle's Chinese Room	106
Georges Perec's <i>Maschine</i>	109
Enzensberger's and Schmatz's / Czernin's poetic machines	111
Software dystopia: Jodi	112
Software dystopia: Netochka Nezvanova	114
From dystopia to new subjectivity	118
Chapter 5. What Is Software?	121
A cultural definition	121
Software as practice	122
Software versus hardware	123
Conclusion	124
References	127
List of Figures	133
Index	135

CHAPTER 1

Introduction: In Dark Territory



FIGURE 1. “300,000 pages of code. Or 60 minutes of triple-X rubber-and-leather interactive bondage porno. Technology can be used for beauty, or debasement. And until you plug it in, you’ll just never know.”

The 1995 Steven Seagal action film *Under Siege 2* tells of an elaborate flow of codes: Villain Travis Dane (Eric Bogosian) hijacks a train and puts a CD-ROM with missile launch codes into a computer to assume control over a global, satellite-based weapon system and blackmail the U.S. government. He trades binary access codes for extortion money, money that itself is digital zeros and ones flowing around the globe to offshore bank accounts. A phantasm of codes as an omnipotent force rules the hijacked train. Seagal’s character, one man army Casey Ryback, and his Apple Newton pocket computer (which sends out a critical fax message to the U.S. army), embody the anti-phantasm. Ryback stands for old-fashioned physics battling symbolic code wizardry, hardware against software. When Ryback kills Dane in the end and a train crash cuts off the satellite link, physics wins over logic. It is furthermore the victory of one genre within the

film over another, fistfighting realism over utopian techno imagination, just like in every fantastic action film from James Bond to *The Terminator* where villain science fiction technology is doomed to be destroyed in the end.

This booklet attempts to show that algorithmic code and computations can't be separated from an often utopian cultural imagination that reaches from magic spells to contemporary computer operating systems.¹ "300,000 pages of code. Or 60 minutes of triple-X rubber-and-leather interactive bondage porno. [...] And until you plug it in, you'll just never know." This dialogue line sums up utopian and dystopian imagination reaching from omnipotence to obscenity projected onto computer codes. In the end, the decoding of the codes is not a formal, but a subjective operation. Boiling down to either "beauty" or "debasement," two classical modes of aesthetics since 18th century philosophy, these codes are ultimately about human perception and imagination.

The science fiction of the film scene relies on a gap between the computer code and a meaning made up by the human viewer. This meaning can't be perceived until the initial code has been transformed several times, from the zeros and ones on the CD-ROM to, for example, pixels on a video screen and eventually a "triple-X rubber-and-leather interactive bondage porno" image in the mind of the spectator. The wider the gap between code and perception, the wilder the imagination. The more abstract a code, the more speculative the meaning that may be read into that code. Long before Steven Seagal, codes stirred up cultural imagination just because they were open to any reading. Western culture believed Egyptian hieroglyphs to hold divine powers until the Rosetta translation stone, found by Napoleon's army in the early 19th century, debunked them as ordinary writing. Hieroglyphs on Freemasonic buildings and documents are a remnant of the older belief. The 16th century *Voynich Manuscript*, written in an as yet unknown alphabet, unknown language and containing obscure pictorial illustrations, has today not yet been deciphered although many expert cryptographers have tried. It is not even clear whether the manuscript contains a cypher at all. It might have been crafted just to create the illusion of a cryptogram. According to other theories, it might be written in a private Thai or Vietnamese alphabet, or by Cathar heretics in a mixture of Old French

¹And which includes the system this paper was written on: the TeX typesetting system, the vi editor and the GNU utilities, each designed by one major speculative thinker of software culture, Donald Knuth, Bill Joy and Richard Stallman respectively.

and Old High German. Artistic speculations on the *Voynich Manuscript* include a story by science fiction writer Colin Wilson who links it to Lovecraft's *Necronomicon*. In a contemporary orchestra piece by Swiss-German composer Hanspeter Kyburz, it serves as a musical score that anticipates 20th century experimental score notations of John Cage and others.

As speculative codes, Egyptian hieroglyphs (in their two different historical readings), the Voynich Manuscript and Travis Dane's CD-ROM render "code" ambiguous between its traditional meaning of a cryptographic code, i.e. a rule for transforming symbols into other symbols, and code in its computational meaning of a transformation rule for symbols into action. Ever since computer programmers referred to written algorithmic machine instructions as "code" and programming as "coding," "code" not only refers to cryptographic codes, but to what makes up software, either as a source code in a high-level programming language or as compiled binary code, but in either case as a sequence of executable instructions. With its seeming opacity and the boundless, viral multiplication of its output in the execution, algorithmic code opens up a vast potential for cultural imagination, phantasms and phantasmagorias. The word made flesh, writing taking up a life of its own by self-execution, has been a utopia and dystopia in religion, metaphysics, art and technology alike. The next chapters will reconstruct the cultural and imaginative history of executable code. From magic spells to contemporary computing, this speculative imagination has always been linked to practical—technical and artistic—experimentation with algorithms. The opposite is true as well. Speculative imagination is embedded in today's software culture. Reduction and totality, randomness and control, physics and metaphysics are among the tropes it is obsessed with, often short-circuiting their opposites. Computer users know these obsessions well from their own fears of crashes and viruses, bloatware, malware and vaporware, from software "evangelists" and religious wars over operating systems, and their everyday experience with the irrationality of rational systems. After all, "until you plug it in, you'll just never know."

CHAPTER 2

Computations of Totality

Exe.cut[up]able statements

```
Date: Tue, 14 Jan 2003 21:47:42 +1100
From: mez <netwurker@HOTKEY.NET.AU>
To: WRYTING-L@LISTSERV.UTORONTO.CA
Subject: Re: OPPO.S[able].I.T[humbs]ION!!
```

```
Hello Arch.E.typal T[Claims of the n]ext W[h]orl.d
----- (mo.dueling 1.1 )-----
```

```
N.terr.ing the net.wurk---
::du n.OT enter _here_ with fal[low]se genera.tiffs + pathways poking
va.Kant [c]littoral tomb[+age].
::re.peat[bogging] + b d.[on the l]am.ned.
::yr p[non-E-]lastic hollow play.jar.[*]istic[tock] met[riculation.s]hods
sit badly in yr vetoed m[-c]outh.
```

```
Pr[t]inting---
::spamnation. .r[l]u[re]ins. .all.
```

```
Exe.cut[up]able statements---
::do knot a p.arse.r .make.
::reti.cu[t]lla[ss]te. yr. text.je[llied]wells .awe. .r[b]ust.
```

```
R[l]un[ge]ning the pro.gram[mar]---
::re.a[vataresque]ct[ors|actrestles] + provoke @ yr response per[b]il[e].
::con.Seed.quenches r 2 b [s|w]allowed.
::big boots make filth k.arm[N limb.ic cyst.M]a.
```

A hybrid of net art, poetry, program and markup code, this piece by Australian net.artist mez (Mary Anne Breeze) reflects a contemporary imagination of software, computation and networks, disassembling it into its smallest symbolic particles and reassembling them into a private language. (*mo.dueling 1.1*) reads as the name of a computer program with a version number. Through its pointed and bracketed word fragments it expands, like running software code, into the words “module,” “duel” and “dueling.” The syntactical notation is taken from wildcards and regular expressions in programming languages and Unix command line interpreters forming an archetypical world. In mez’s own language “mezangelle,” it is called “Arch.E.typal

T[Claims of the n]ext W[h]orl.d,” expanding into hardware/software architectures—“arch” in computer tech lingo—, “claims of the next world,” and whores. Slang and sexual language exposes mezangelle as a messy code, one that does not run on machines, but on a human imagination that encompasses machines and bodies alike. Unlike in the Steven Seagal action movie and the Voynich Manuscript, the semantic associations are not superimposed and therefore external to the code, but are embedded into the code proper. “Exe.cut[up]able statements” and “pro.gram[mar],” for example, are self-reflections of the text as “executable statements” and “program grammar.” The words serve as a source code which generates “execute,” “executable,” “cut-up,” “able” in the first word, “programmer,” “grammar,” “program,” “gram” in the second. There is simultaneous contraction and expansion, regularity and irregularity, instruction and chaos in these words. Like a piece of software code that gets executed, the writing expands beyond itself, generating dozens to hundreds of possible readings. As it says, its “big boots make filth,” and text disperses in a “spamnation.” “Spamnation” also is a technical description of mez’s E-Mail work that is sent, spam-style, to a large number of net cultural mailing lists at once.

From a literary history viewpoint, the word hybrids and onomatopoeics of mezangelle resemble the poetic language in James Joyce’s last novel *Finnegans Wake*:

The fall (bababadalgharaghtakamminarroninikonn-
bronnontonnerronnntuonnthunntrovarrhounawnskawnto-
ohoohoordenenthurnuk!) of a once wallstrait oldparr
is retaled early in bed and later on life down through
all christian minstrelsy. The great fall of the offwall
entailed at such short notice the pftjschute of Finnegan,
erse solid man, that the humptyhillhead of himself
promptly sends an unquiring one well to the west
in quest of his tumptytumtoes: and their upturnpike-
pointandplace is at the knock out in the park where
oranges have been laid to rust upon the green since
devlinsfirst loved livvy. What clashes here of wills gen
wonts, oystrygods gaggin fishygods! Brékkek Kékkek
Kékkek Kékkek! Kóax Kóax Kóax! Ualu Ualu Ualu!
Quaouauh!

This paragraph, from the first page of the book, demonstrates Joyce’s word poetics while referencing its literary prototypes. The onomatopoeic “Bkkek Kkkek [...]” is taken from Aristophanes’ 4th

century B.C. Greek comedy *The Frogs*. “Humptyhillhead,” “prumptly” and “tumptytumtoes” allude to Humpty Dumpty, the nursery rhyme character and fantastic creature in Lewis Carroll’s *Through the Looking-Glass*, the sequel to *Alice in Wonderland*. In its sixth chapter, Alice and Humpty Dumpty discuss the seemingly nonsensical poem *Jabberwocky*:

’Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Humpty Dumpty explains that “slithy” is a combination of “lithe and slimy”: “‘Lithe’ is the same as ‘active.’ You see it’s like a portmanteau—there are two meanings packed up into one word.” Originally, and before it appeared in the novel, the “Jabberwocky” poem had been written in 1855 and published under Carroll’s proper name Charles L. Dodgson as a parody of romantic poetry. Dodgson was a mathematician by profession and taught at Oxford. As Martin Gardner’s annotated edition of the two Alice novels shows,—which also became published as one of the first electronic hypertext books ever—the books are rich with mathematical and logical puns and humor. In the “Humpty Dumpty” chapter, absurdity stems from a seemingly straightforward, pseudo-logical explanation of the nonsense poem. Humpty Dumpty takes the attitude of a mathematician or logician who reads the poem like a formula. Joyce’s portmanteau words hybridize languages, nature and history, but while they hardly ever mix in formal or machine language, they work as a kind of reverse computer code that expands multiple input into a single output. His novel is infinitely looping, with its last page of the novel ending where the first page begins. Marshall McLuhan, “Joycean hippie” (as Nam June Paik called him) and founder of the term “media theory,” took *Finnegans Wake* as his “textbook.”¹ According to his biographer Donald Theall, he perceived the blend of “orality, tactility, simultaneity and synaesthesia” in the novel as a blueprint for a “techno-poetic” language.²

Having created, more or less, the field of “media” with its endless unresolved terminological ambiguities and contradictions, McLuhan’s techno imagination appears to bridge the gap between the technopoetic codework of mez and her net contemporaries on the one hand

¹According to http://www.geocities.com/hypermedia_joyce/theall13.html

²According to <http://www.mindjack.com/feature/mcluhan.html>

and its prototype in Carroll and Joyce on the other.³ “Spamnation,” code flowing out and spilling over, is the common denominator of Carroll’s “slithy toves,” Joyce’s “riverrun” and mez’s collapsing of program, grammar and programmer into a “pro.gram[mar]”. All three artists write in the aesthetic mode of the sublime; the category of the boundless, unshapely, obscure, threatening first described in the Greek rhetorical treatise of Pseudo-Longinus, reinvigorated in the 18th century aesthetic theories of Burke, Kant and Schiller,⁴ and which paved the ground for the gothic novel and other forms of dark romanticism (up to the gothic, dark wave and “new romantic” pop cultures that originated the 1980s). Computer and software imagination is only inscribed into mez’s piece. The software her coding conceives of is a monster, an alien resembling that of the eponymous Science Fiction movies. Unlike the “cyberpunk” Science Fiction imagination of the late 1980s and early 1990s whose sublime technopoesis consisted of an imaginary pictorial hyperrealism, mez’s monsters are made up purely by abstract symbols and computational processes.

Magic and religion

Words becoming flesh, the symbolic turning physical – these are by no means recent phantasmagorias and speculations. The beginning of the Gospel of John in the New Testament reads:

1:1 In the beginning was the Word, and the Word was with God, and the Word was God. [. . .]

1:14 And the Word was made flesh, and dwelt among us, (and we beheld his glory, the glory as of the only begotten of the Father,) full of grace and truth.

This figure of thought, of a speech act that affects physical matter instantly and directly, is magical in its root. Material creation from the word is an idea central to magic in all cultures; it is precisely

³Outside the Anglo-American tradition, the word recombinations and cosmological imagination of Russian futurist poet Velimir Khlebnikov manifests another pretext which itself triggered the quasi-computational structuralist poetics of Roman Jakobson

⁴W. Rhys Roberts, editor. *Longinus on the Sublime*. Cambridge University Press, Cambridge, 1899. [85], Edmund Burke. *A Philosophical Enquiry into the Origin of our Ideas of the Sublime and Beautiful*. Oxford University Press, Oxford, 1990 (1757). [15], Immanuel Kant. *Critique of the Power of Judgment*. The Cambridge Edition of the Works of Immanuel Kant in Translation. Cambridge University Press, Cambridge, 2001. [53], Friedrich Schiller, *On The Sublime*, <http://members.aol.com/abelard2/schiller.htm>

what magic spells perform. Magic therefore is, at its core, a technology, serving the rational end of achieving an effect, and being judged by its efficacy. According to scholar Franz Dornseiff and his 1922 German study on the alphabet in mysticism and magic,⁵ the idea of divine creation through the letter has its roots in early Middle Eastern and Egyptian mystic cults. Gnosticism transformed it into theurgy, the invocation of divine powers for achieving concrete, material effects. Adopting many of its concepts from Gnosticism and Neoplatonist philosophy, Christianity introduced the prayer as its own form of theurgy, itself a practical communicative act between the individual, the divine and physical matter through a symbolic agent, or, medium. Magical thinking is even more strongly present in the Catholic Christian idea of transubstantiation, the transformation of wine into blood and bread into the body of Christ effected through the liturgic speech act of the priest.—The magical formula “hocus pocus” is derived from the Catholic liturgic formula “hoc est corpus meum,” this is my body.⁶—Rationalization of this remnant of magical thinking occurred within Christianity itself when Protestantism abandoned the concept of transubstantiation and thought of communion service as a purely allegorical practice.⁷

Religion sublimates magic as a common, popular into a privilege of the god creating the world and, subsequently, his son. Magic wasn't considered occult until religion and later science and technology rivalled and marginalized it. The technical principle of magic, controlling matter through manipulation of symbols, is the technical principle of computer software as well. It isn't surprising that magic lives on in software, at least nominally. References to magic abound in computer software branding, from programs like *Partition Magic*, *Magix Musicmaker*, the */etc/magic* filetype database in Unix to the program genre of “Setup Wizards” or operating systems like *Sorcerer GNU/Linux* in which software packages are installed with the command “cast.” A Google search on “magic” and “software” today yields more than fifteen million results (see figure 1). Searching the word

⁵Franz Dornseiff. *Das Alphabet in Mystik und Magie*. Teubner, Leipzig, Berlin, 1925. [29].

⁶This link was first made by English prelate John Tillotson in 1742, according to *The American Heritage Dictionary of the English Language*

⁷A rationalization that has been progressively extended in non-evangelical Protestant theology to practically all miracles and para-magical acts described in the Bible, most or all of which were subsequently declared allegorical.

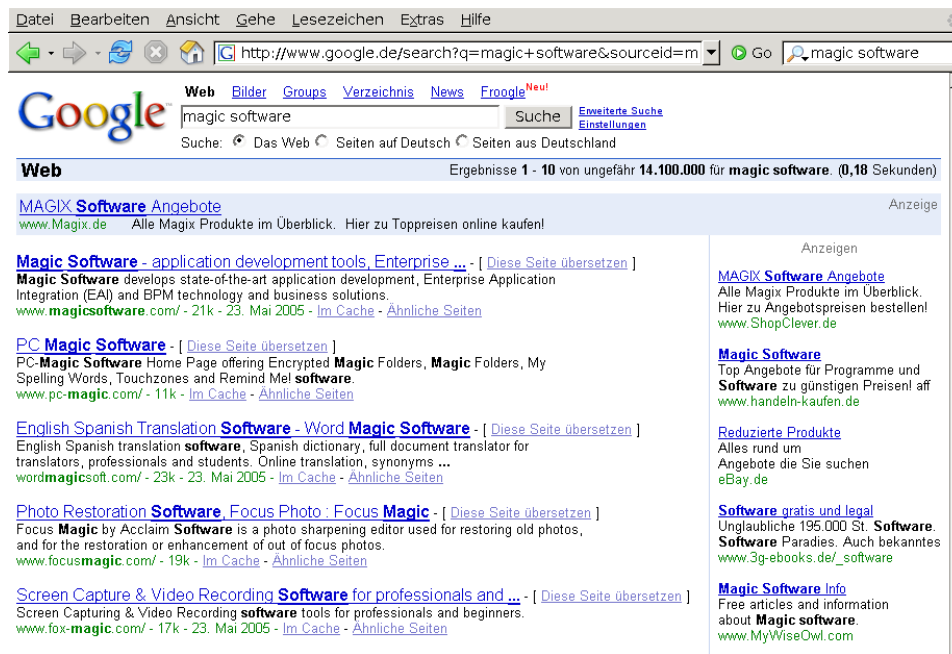


FIGURE 1. Result of a Google search on “software” and “magic”

“magic” only brings up the homepage of a software company as the third result.

Such links between magic and software remain metaphorical though unless they are based on common concepts of formalism, language and execution of statements. Since magic was marginalized in Western culture through science, it went underground. While magical practices as those of Aleister Crowley claimed to be “scientific,” they doubtlessly existed outside a system of science and rationality which before, until roughly the late 17th century, still had included occult science in its canon. Of the “two cultures” described by scientist and novelist C.P. Snow in 1956, hard sciences and engineering versus humanities and culture, Crowley’s magic clearly sides with the second despite its scientific claims, de-emphasizing magic as a practical technique in favor of magic as an occult philosophy of life. His influence was strong particularly in pop and non-mainstream culture, forming one philosophical formulation of the underground *per se*. Followers of Crowley include experimental filmmaker Kenneth Anger, the industrial music movement founded by Genesis P-Orridge and others, The Rolling Stones whose song *Sympathy for the Devil* was Crowley-inspired and The Beatles who put Crowley on the cover of *St. Pepper’s Lonely Hearts Club Band*.

Magic as Crowleyan occult philosophy, art and programming converged in the poetic language experiments of Brion Gysin and William S. Burroughs:

IN THE BEGINNING WAS THE WORD
 IN THE BEGINNING WAS WORD THE
 IN THE BEGINNING WORD THE WAS
 IN THE BEGINNING WORD WAS THE
 IN THE THE BEGINNING WAS WORD
 IN THE THE BEGINNING WORD WAS
 IN THE THE WAS BEGINNING WORD
 IN THE THE WAS WORD BEGINNING
 IN THE THE WORD BEGINNING WAS
 IN THE THE WORD WAS BEGINNING
 IN THE WAS BEGINNING THE WORD
 IN THE WAS BEGINNING WORD THE
 [...] ⁸

The poem shuffles its words according to a formal algorithm. Its total of 720 permutations were calculated, in the early 1960s, on a Honeywell computer with the aid of mathematician Ian Sommerville. Spoken by the author on a tape recording, this and other permutation poems of Gysin were not solely mathematical computations, but also incantations. It therefore does not seem incidental that the beginning of the Gospel of John was chosen to be computed. The power of creation in the word is being resurrected in the text from a previously hidden potential. Combinatory computation turns into the technical agent, or spell of this magic act, but its power is ultimately embodied by the speaking voice. The principle of text cut-ups, as they were developed and practiced by Gysin and Burroughs (see p. 76) is contained also in the permutational poems since they work with the same principle of slicing out and shuffling portions of a text: lines in Gysin's poems, columns in Gysin's and Burroughs' cut-up prose. In his essay, *Cut-ups self-explained*, Gysin argues that "Words have a vitality of their own." Through permuting them, he writes, one can make them "gush into action." The result is an "expanding ripple of meanings which they [i.e. the words] did not seem to be capable of when they were struck into that phrase."

Likewise, William S. Burroughs' essay *The Electronic Revolution*, which theoretically sums up, although not very coherently, his cut-up poetics, begins with the sentence: "In the beginning was the word

⁸Brion Gysin. Permutation poems. In: William S. Burroughs. *The Third Mind*. Viking, New York, 1978. [42].

and the word was god and has remained one of the mysteries ever since. The word was God and the word was flesh we are told. In the beginning of what exactly was this beginning word? In the beginning of WRITTEN history.”⁹ In *The Electronic Revolution*, written in 1970, the tape recorder takes the place of the technical agent for dissecting and reassembling language and cultural codes for which Gysin employed the computer. But for both writers, technology is a formal means to ultimately manipulate meaning, not formalisms, whereas a computer program, be it a compiler, a word processor, an image or sound manipulation program, is conceived of as a symbolic formalism employed for an equally formal manipulation of symbols. Burroughs’ and Gysin’s insistence on the cultural power of speech acts and media manipulation instead aims at a programming that is semantic, not formalist. So they transfigure the cut-up method from a formalism to a technique that is semantic *in itself*, being an occult and magical practice likened to drugs and ecstatic experience.

Another such technical device, Gysin’s *Dream Machine*, is a visual flicker generator based on a visual pattern cut into a cylindrical piece of cardboard rotating on a record player. The visual frequencies it generates can interfere with brain frequencies and create optical illusions, a psychedelic technology used also in Tony Conrad’s experimental films from the 1960s, and with roots in military behaviorist research. That technology is semantic, ecstatic and magic is a point made also in an 1984 underground movie aptly called *Decoder* which stars Burroughs and Genesis P-Orridge in cameos and whose accompanying *Decoder Handbuch* [*Decoder Handbook*] includes texts by him and Crowley as background material.¹⁰

According to anthropologist James George Frazer, it is a general feature of magic that magical practices tend to cloud their technical and formalist nature, enmeshing themselves with the semantics of the objects and subjects they are intended to affect. In his study *The Golden Bough* (1922), Frazer differentiates two types of magic, imitative and contagious magic. In imitative magic, the action or effect to be achieved is being mimetically reproduced, for example in the Voodoo-like “attempt which has been made by many peoples in many

⁹William S. Burroughs. *Electronic Revolution*. Expanded Media Edition, Bonn, 1982. [19].

¹⁰Klaus Maeck and Walter Hartmann, editors. *Decoder Handbuch*. Trikont, Duisburg, 1984. [64]

ages to injure or destroy an enemy by injuring or destroying an image of him.”¹¹ Contagious magic, on the other hand, works through physical proximity, and is best exemplified, according to Frazer, by “the magical sympathy which is supposed to exist between a man and any severed portion of his person, as his hair or nails; so that whoever gets possession of human hair or nails may work his will, at any distance, upon the person from whom they were cut.”¹²

Both operations can be fundamental operations in language and art as well. Roman Jakobson, a founding theorist of 20th century linguistic and literary structuralism, made Frazer’s concept of imitative magic—or the principle of similarity—the basis of a new definition of metaphor, and contagious magic—or the principle of proximity or contiguity—the foundation of metonymy.¹³ In commercial computer programs like *Partition Magic*, the relation between software and magic is metaphorical in this sense of imitation and similarity; the software is sought to achieve magical powers by referring to them only nominally, putting them onto its label rather than into its code.

In Gysin’s and Burroughs’ cut-up poetics however, the technical process renders itself magic not just metaphorically, but physical by inclusion. The opening phrase of the Gospel of John is a prime example. Both the permutation poem *IN THE BEGINNING WAS THE WORD* and *The Electronic Revolution* contain the magical spell of the Gospel not as an allusive imitation, but as a physical inscription and contagious agent, just like the scalp of an enemy ripped out and worn on the body as an amulet. Through the cut-up process, the words regain their original contagious quality and “gush into action,” as Gysin puts it. Burroughs literally links cut-ups and linguistic contagion in what is perhaps his best-known speculation—that language is a virus. If language is a virus, then cut-up literature is about unleashing and applying its viral potential. For Burroughs, the affinity of language and viruses is quite literal. It amounts to more than the idea that viruses could be created in language or, like in Richard Dawkins’ concept of the “meme,” that certain speech acts had contagious effects.¹⁴ Burroughs stresses that

¹¹James George Frazer. *The Golden Bough*. Macmillan, London, 1950. [35], p. 12

¹²*ibid.*, p. 38

¹³Roman Jakobson. Two Aspects of Language and Two Types of Aphasic Disturbances. In *Fundamentals of Language*, pages 115–133. Mouton, The Hague, Paris, 1971. [50]

¹⁴Richard Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, 1989 (1976). [28]

I have frequently spoken of word and image as viruses or as acting as viruses, and this is not an allegorical comparison.¹⁵

Burroughs' language magic is contagious in a double respect. It has a contagious effect and it is contagious in its very structure. It therefore differs from Frazer's contagious magic in which the contact is firstly limited to one object and its bearer, and secondly not inscribed into the system of signs itself. With his phantasmagoria of all-pervasive infection that cannot be contained, Burroughs totally semanticizes the mobilization of matter through symbols in the magical speech act. This corresponds, oddly and in a non-canonical way, with older philosophical concepts of codes permeating the cosmos.

Pythagorean harmony as a cosmological code

Pythagorean musicology. Pythagorean thinking is founded on the idea that the world is organized in numerical proportions which are coded equally into music and mathematics. There exists no record of Pythagoras' original philosophy though because it existed only as secret knowledge within an occult society. Heraclitus is one of the earliest purveyors of Pythagorean ideas: "Combinations, wholes and not-wholes, conjunction and separation, concord and discord—out of all things comes One, and out of One all things," or, in a different translation: "What goes against each other is joining; what strives apart creates the most beautiful harmony."¹⁶ When Pythagoras discovered the arithmetic principle of the musical octave by splitting the string of a monochord in half, and from that concluded that there was a mathematical harmony of the cosmos, he founded an aesthetic philosophy that closely linked art, science and nature and whose impact was immense through the Renaissance and beyond.

Pythagorean aesthetics centers around the idea that beauty is made up from mathematical proportions. The mathematical proportions of sound were first described with the numbers 3-4-6 as the relative numerical values of keynote, quint and octave. This code was written down in the treatise *De musica* of Anicius Manlius

¹⁵William S. Burroughs. *Electronic Revolution*. Expanded Media Edition, Bonn, 1982. [19], p.59

¹⁶Jonathan Barnes, editor. *Early Greek Philosophy*. Penguin, Harmondsworth, 2002. [7]

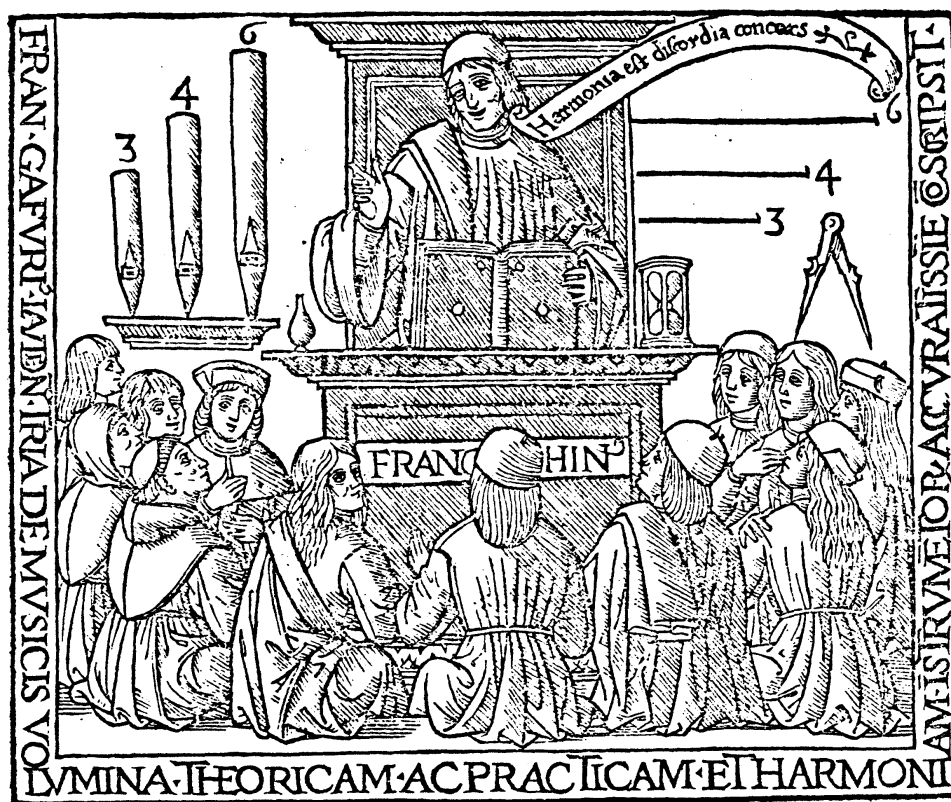


FIGURE 2. Franchino Gaffurio, *De Harmonia Musicorum* (1518)

Boethius, a 6th century B.C. Latin philosopher.¹⁷ It remained canonical until the invention of tempered tuning and Mersenne's mathematical musicology (see p. 106) in which intervals smaller than the octave were no longer determined by absolute integer values, but relative to each other. On the title illustration of his 1518 treatise *De Harmonia Musicorum*,¹⁸ the Italian Renaissance composer and musicologist Franchino Gaffurio has himself portrayed teaching Boethius' Pythagorean 3-4-6 code to his pupils. The picture illustrates the equivalence of musical tuning and mathematical values with organ pipes and a pair of compasses. With speech drawn in anticipation of comic strip balloons, Gaffurio tells his pupils "Harmonia est discordia concors," harmony is concordant discord, the Pythagorean credo previously voiced by Heraclitus (figure 2).

¹⁷Boethius. *De Musica*. In *Opera omnia*. Firmin-Didot, Paris, 1882. [11]

¹⁸Franchino Gaffurio. *De harmonia musicorum*. Forni, Bologna, 1972 (1518). [36]

This concept of harmony is very distinct from the modern understanding of the term, because it includes *both* melodics and harmonics. In other words, a melody can, according to this concept, be harmonic as well. Secondly, it integrates dissonance with consonance. This understanding is implied already in the original Greek word “harmonia” which literally means “joining” or “combining.” Beauty, it follows, is not plainly nice and agreeable, but made up of agreeable and disagreeable elements in equal proportions. This is not only the general principle of Pythagorean aesthetics, but also of any refined classicism.¹⁹

Rhetoric of the acumen. In the 17th century, the joining of opposites became a principle of witty or “conceited” poetics and poetry. This literature later became historicized as anti-classicist, “mannerist” or “baroque.” Wit, or “acumen,” was a part of rhetoric and poetics since the 16th century. It became systematically taught in Jesuitical academia. Wit was considered the principle of epigram poetry, and epigrams in turn functioned as the “subscriptio,” explanatory verse underneath pictorial emblems. Emblems, allegorical images, were hugely popular in the Renaissance since Italian humanist scholar Andreas Alciatus published the first emblem book in 1531.²⁰ Just as desktop icons on computer operating systems were invented in the 1970s in the Xerox PARC labs in order to simplify interaction with the computer as a machine performing formal-logical manipulations of coded symbols, emblems served to simplify and popularize interaction with religious and philosophical meaning. They became an important part of humanist education—and of Jesuit counter-reformational propaganda, likewise based on the humanist canon. Seminal treatises on epigrammatic wit, some of them covering also the creation of wit in emblems, were written by the Jesuits Kazimierz Sarbiewski, Baltasar Gracián and Emanuele Tesauro.²¹

¹⁹Like that of Winckelmann and Lessing in the late 18th century in their respective analyses of the Greek Laocoon sculpture, or the aesthetic philosophy of Walter Pater; see Simon Richter. *Laocoon's Body and the Aesthetics of Pain: Winckelmann, Lessing, Herder, Moritz, Goethe*. Wayne State University Press, Detroit, 1992. [84] and Harold Bloom, editor. *Selected Writings of Walter Pater*. Columbia University Press, New York, 1982. [10]

²⁰Andreas Alciatus. *Emblematum Libellus*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1991 (1542). [2]

²¹A comprehensive survey of their work can be found in Mercedes Blanco. *Les rhétoriques de la pointe*. Librairie Honoré Champion, Paris, 1992. [9].

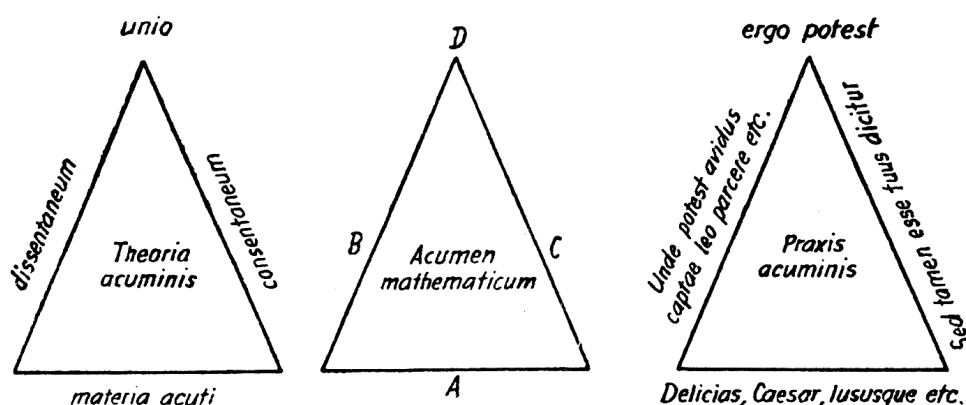


FIGURE 3. Diagram from Maciej Kazimierz Sarbiewski's *De acuto et arguto* (1627)

Kazimierz Sarbiewski, famous in his time as the “Polish Horace,” is of particular interest since his treatise *De Acuto et Arguto*, published in 1627,²² uses a Pythagorean model to sketch a quasi-algorithm for composing wit. He defines acumen, rhetorical wit, as a junction of a converging and a diverging meaning that culminate in a paradoxical union, or, point. This point is explained with the Pythagorean formula used earlier by Gaffurio, “discordia concors.” Just like other Pythagorean thinkers who explain laws of form in the arts and technology through numerical models, Sarbiewski equates rhetorical composition to mathematics and geometry. His rhetorical acumen is juxtaposed to an “acumen mathematicum,” a triangle, and explained with a triangular diagram (figure 3). The base represents the material, left and right arms/sides the elements of dissent and consent respectively, the edge the union, or witty point. So Sarbiewski’s acumen is indebted to the Pythagorean and Renaissance idea of harmony as a product made up of opposites striving against each other. His triangle functions as a computation device for wit. Yet it is not an algorithmic text generator in a strict sense because it presupposes semantics. Unlike in proper computations, Sarbiewski’s wit cannot be computed through a purely formal, syntactical manipulation of symbols.

Another reading of Jesuit acumen rhetoric as computation can be found in Umberto Eco’s novel *The Island of the Day Before*.²³ Its

²²Maciej Kazimierz Sarbiewski. *De Acuto et Arguto liber unicus*. In *Wykladi Poetyki*, pages 1–20. Wydawnictwo Polskiej Akademii Nauk, Wrocław, Krakow, 1958. [87]

²³Umberto Eco. *The Island of the Day Before*. Penguin, Harmondsworth, 1996. [33]

ninth chapter, *The Aristotelian Telescope*, bears the title of Emanuele Tesauro's 1654 acumen treatise *Il cannocchiale aristotelico*.²⁴ Tesauro himself appears in the novel as "Padre Emanuele." His idea of using—not Pythagorean triangles, but—Aristotelian logic as a formal device for synthesizing ingenious metaphors is taken up by the novel and gets transcribed into the fiction of a computing device:

The base consisted of a great chest or case whose front held eighty-one drawers-nine horizontal rows by nine vertical, each row in both directions identified by a carved letter (BCDEFGHIK). On the top of the chest, to the left stood a lectern on which a great volume was placed, a manuscript with illuminated initials. To the right of the lectern were three concentric cylinders of decreasing length and increasing breadth (the shortest being the most capacious, designed to contain the two longer ones); a crank at one side could then, through inertia, make them turn, one inside the other, at different speeds according to their weight. Each cylinder had incised at its left margin the same nine letters that marked the drawers. One turn of the crank was enough to make the cylinders revolve independently of one another, and when they stopped, one could read triads of letters aligned at random, such as CBD, KFE, or BGH.²⁵

Eco's fiction conflates two logical-rhetorical proto-computers with each other, Tesauro's acumen rhetoric with the "ars" of medieval Catalan monk Ramon Llull (see p. 36).²⁶ Even if Tesauro's machine is a modern fiction, it is true that the Jesuit rhetorical manuals on the acumen sought to radically formalize the production of "ingenium,"²⁷ an ingenuity that would be spelled one century later, as "genius." The Renaissance "ingenium" however is not the irregular, anti-methodical genius of romanticism, but an effect that can be synthetically created by anyone. The rise of the romantic genius as the model of an artist

²⁴See Mercedes Blanco. *Les rhétoriques de la pointe*. Librairie Honoré Champion, Paris, 1992. [9]

²⁵Eco, *ibid.*, chapter 9

²⁶Anthony Bonner, editor. *Doctor Illuminatus: A Ramon Llull Reader*. Princeton University Press, Princeton, New Jersey, 1993. [12]

²⁷Gracián's book has the according title *Agudeza y arte de ingenio*. An electronic version of the 1648 book exists at <http://www.cervantesvirtual.com/servlet/SirveObras/12259950118927841194513/>

who does not calculate in turn explains the crisis of computational methods and games in poetry and arts between ca. 1800 and 1900.

Continuity of Pythagorean thinking. The Pythagorean project consists of the extraction and application of a universal numerical code that organizes both nature and art. This code allows the creation of a correspondence between macrocosm and microcosm and describes harmony, in the sense of beautiful numerical proportions, as the guiding principle of the world. And for the first time, it allows the computation of nature and art. Any natural and symbolic system can be broken up into numerical proportions and values which in turn may be compared to the numerical proportions and values of another observed system. It is this principle of universal similarity and correspondence which Eco calls the “hermetic paradigm” and sums it up under the maxim “sicut superius sic inferius,” “as above, so below” to describe a correspondence of macro- and microcosm.²⁸ In Pythagorean and later hermetic thinking, numerical proportions can be universally equated to geometrical proportions and musical intervals. Letters, likewise, can be computed as numbers and set into relation to the numerical intervals which are thought to be the foundations of the cosmos. Pythagorean thought therefore first coined and systematically expressed the idea that a symbolic-mathematical source code underlies the universe and describes nature and culture alike.

The Pythagorean tradition anticipates the modeling of culture through software, very literally in the case of music. The law that an octave is a division of a frequency into its half, for example, is implemented into every sound synthesis computer program in the world. Computer-generated music and computer-generated digital instrument programming remains the most systematic elaboration of the original Pythagorean project of finding numerical and arithmetic models for sound. The models reach from simple algorithms like the division of the octave to more recent and complex ones such as Fourier overtone frequency transformations, originally put down by the French mathematician Jean Baptiste Joseph Fourier in 1822. The relative success of mathematical modeling of music explains why music is the oldest and formally most advanced type of computer-based art. Algorithmic musical composition software such as *MAX*

²⁸From his book *Streit der Interpretationen* which, so far, has only been published in German, Umberto Eco. *Der Streit der Interpretationen*. Universitätsverlag Konstanz, Konstanz, 1987. [30]

and *Pure Data* by Miller Puckette has grown into a number of complete, self-contained programming environments and user interfaces whose application is no longer restricted to audio processing.²⁹

Twentieth century avant-garde music composition technique forms a major link between Pythagoras' discovery of mathematical laws in music and music composition software like *Pure Data*. The dodecaphonic music of Arnold Schönberg was based on the principle of permuting the order of the twelve halftones of an octave. All twelve halftones are treated equally, resulting in a composition with no tonality and no tonal center such as "C major." The techniques used to permute twelve tone rows, retrograding and numerical inversion for example, were in fact algorithms. In his late work, Anton Webern expanded the permutation row principle to all musical parameters, not only pitch, and laid the ground for the serial music of Karlheinz Stockhausen and Pierre Boulez in the 1950s and 1960s.³⁰ The latter sought to scientifically define the parameters of sound, organizing composition according to permutation rows and their complex polyphonic variation. Twentieth century compositional music thus reappraised the 17th century music of Johann Sebastian Bach. Bach's composition techniques particularly in *The Goldberg Variations*, *The Art of the Fugue* and *The Musical Offering* brought the Renaissance Pythagorean concept of music to its highest and most rigorous formal level—which explains its fascination for present-day computer programmers and artificial intelligence theorists like Douglas R. Hofstadter in *Goedel, Escher, Bach*.³¹

When serial composition in the 1950s and 1960s put all sound parameters under the regime of numerical permutation, musical composition became thoroughly computational. Yet most its computations were still calculated by hand. From this, it was only a short route to analog electronic sound generation, like in Stockhausen's *Studie II*, and computer-aided composition. In 1977, Pierre Boulez founded and directed the IRCAM electronic music center in Paris where seminal audio software like MAX was developed. For Stockhausen, total numerical control over sound eventually led to a historical reversion

²⁹See Miller Puckette's homepage <http://www-crcra.ucsd.edu/~msp/software.html>

³⁰Stockhausen's analysis of an orchestra piece by Webern laid the groundwork for serial composition: Karlheinz Stockhausen. *Weberns Konzert für neun Instrumente op. 24*. In *Texte zur elektronischen und instrumentalen Musik*, pages 24–31. M. DuMont Schauberg, Köln, 1963 (1953). [96]

³¹Douglas R. Hofstadter. *Gödel, Escher, Bach*. Basic Books, New York, 1979. [46]

of the secularization process that began with Pythagoras' occult cosmology and ended with modern mathematics and computing. In the course of the 1960s, Stockhausen turned from rationalism to mysticism, arriving at the idea that the cosmos exists through harmonic waves that permeate all matter and existence. The result is a total musicality of the universe. At the same time, Stockhausen became subject to harsh criticism by anti-art activist and philosopher Henry Flynt who in the 1960s was associated with Fluxus³² and by his own former student and assistant, the English composer Cornelius Cardew. Independently from each other, Flynt and Cardew attacked Stockhausen as an "imperialist" and suprematist of the Western musical tradition, and, by implication, its Pythagorean heritage.

The idea that beauty materializes in numerical proportions according to mathematical laws continues to be popular in scientific and engineering cultures, too. Since the early 1970s, Donald Knuth, widely considered the founder of computer science as an independent academic discipline, published his textbooks under the title *The Art of Computer Programming*.³³ He understands "art" as the formal beauty and logical elegance of the source code. The software *TeX* which he wrote to typeset his books correspondingly implements a classicist post-Renaissance typography whose notions of beauty are embedded in Knuth's algorithms for line spacing and paragraph adjustment. At MIT, Knuth initiated a project *God and computers* whose results were an exhibition of Bible calligraphies and, in 2001, a book *Things a Computer Scientist Rarely Talks About*.³⁴ In this book, Knuth remembers how as a student he read a computer program code that he found "absolutely beautiful. Reading it was just like hearing a symphony."³⁵ This was how he "got into software," teaching it as an art rather than a science. The hacker credo put down by Steven Levy in 1983 that "you can create art and beauty with computers" has its roots in Knuth's teaching.³⁶ It ultimately means that a program is not a transparent tool for creating beauty—like, for example, a graphics program—, but that it is beautiful by itself. Both schools, highbrow academic computer science and more underground hacker culture, perpetuate a Pythagorean, classicist understanding of art as formal

³²Most of Flynt's writings are online under <http://www.henryflynt.org>

³³Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, 1973-1998. [55]

³⁴Donald E. Knuth. *Things a Computer Scientist Rarely Talks About*. CSLI Publications, Stanford, 2001. [56]

³⁵Donald Knuth, *Things a Computer Scientist Rarely Talks About*, p. 130

³⁶Steven Levy. *Hackers*. Project Gutenberg, Champaign, IL, 1986 (1984). [63]

beauty. This concept blatantly lags behind modern concepts of art. Since romanticism and 20th century art, aesthetic understandings of art were not just about beauty, but included the sublime, grotesque and ugly as well. The same is true, implicitly at least, for the Greek and Roman antiquity whose highest art form, tragedies, were about violence and despair.

Fractal geometry and the widespread aesthetic fascination for Mandelbrot sets are another example of contemporary Pythagorean aesthetics. They prove that mathematical concepts of beauty do not necessarily have to be founded on strict order and regular proportions, but can also be built on chaotic iteration (see p. 50). The idea, after all, is the same; that the cosmos can be mathematically decoded and art realizes itself as a formal beauty that includes nature and culture alike. Moreover, the “chaos” in the fractals is relative—and not an ontological chaos, i.e. it is not unpredictable—as it is, first of all, framed within a finite surface of a graphical screen, and secondly the fractal forms generate subjectively repetitive and hence predictable patterns (see p. 77). *The Beauty of Fractals*—the title of a book by Hans Otto Peitgen which popularized Mandelbrot graphics in 1986—³⁷ boils down to a static beauty just like that of the Pythagorean fixed proportions from the octave to the golden section.

The Pythagorean tradition, and with it the contemporary aesthetic thinking in hard science and engineering cultures, understands correspondences between art and mathematics in terms of numerical “harmony.” These numbers can be considered a code and formal language, but they are, in the classical Pythagorean model, not yet a source code and language that instigates processes. There is code, but no execution in the code. Magic, on the other hand, lacks the concept of a formal language just because it conflates, through its two modes of similarity and contact, its own language with the objects and subjects involved in its act. Gysin and Burroughs still follow this tradition when they fashion computers and tape recorders into occult ecstatic devices and transform gramophones into “dream machines.” Language magic thus conceives of execution, but typically does not have a mathematical understanding of itself.

With their respective corresponding deficiencies, magic and Pythagorean thinking are two prototypes of programming and software, the former lacking a rigorous concept of abstract mathematical symbol language, the latter lacking a concept of executing symbols.

³⁷Hans Otto Peitgen. *The Beauty of Fractals*. Springer, Heidelberg, New York, 1986. [73]

The history of software begins, in Western cultures at least, where magic and Pythagorean thinking coincide.

Kabbalah

In Hebrew, letters and numbers are mapped into the same notation of the “alefbet.” Unlike Latin where only a small subset of the alphabet corresponded to numbers, every Hebrew letter is also a number. In combination with the idea of divine creation out of the word, or letter, this amounted in the Middle Ages to a complex system of Jewish mystical letter computations. The Kabbalah effectively combined the Pythagorean idea of the world being composed of numbers so that everything can be described in numerical terms and proportions, with the magical concept of language as an agent that affects matter. The religious sublimation of this concept exists in the idea that God created the world through language and that even humans possessed the power of influencing things through the Adamic language spoken in paradise. Kabbalah may be regarded a speculative science of reconstructing the grammar and vocabulary of the Adamic and divine language—through among others computational and pythagorean-numerological readings of the Torah. In its practical application, magical, respectively theurgic, acts are performed. In 1800, the Jewish German rationalist philosopher and Kant scholar Solomon Maimon published an autobiography in which he recollects his juvenile Talmudic and Kabbalah studies in Lithuania.³⁸ He defines Kabbalah almost linguistically as “the doctrine teaching how to willingly affect nature by means of the manifold names of God which represent specific modes of working upon, and relations to, natural objects. These names are regarded not just as arbitrary, but as natural signs so that everything done with these signs affects the objects they represent.”

As in magical language, Kabbalist mystics conceive of letters and words not merely as abstract, artificial denominators which are culturally defined and shared. They thus stood in opposition to rationalist definitions of language in Aristotelian philosophy, medieval nominalism and the late 19th century structuralist linguistics of Ferdinand de Saussure which all insisted upon the constructedness of language. Since Kabbalah is mystical scholarship aimed at retrieving the original power of the divine letters and words, it also aims at applying this power practically. Modern Kabbalah scholarship, including that

³⁸Solomon Maimon. *An Autobiography*. University of Illinois Press, Chicago, 2001 (1792). [65]

of Gershom Scholem, has obscured this end by focusing solely on theoretical Kabbalah. Today, practical Kabbalah includes the politics of the ultraorthodox Shas party in Israel and, on opposite fronts, the popularized, no more exclusively Jewish New Age Kabbalah of the *Kabbalah Centre* whose adepts include Madonna, Liz Taylor, Elton John, Mick Jagger, Courtney Love and Britney Spears. In its historical form, practical Kabbalah is theurgy, i.e. magical rituals performed within religion as invocations of god. Maimon remembers his own attempts at practical Kabbalah as follows:

With the Kabbalah Ma'asith or practical Kabbalah, I did not not succeed so well as with the theoretical. The preacher boasted, not publicly indeed, but to everybody in private, that he was a master of this also; especially he professed roeh ve-eno nireh (to see everything, but not to be seen by others), i.e., to be able to make himself invisible. I was particularly eager for this artifice in order that I, a young person, might practice certain kinds of mischief on my comrades without being punished. [...] Three days in succession I had to fast, and every day to say some Ichudim. These are Kabbalistic phrases of prayer whose occult meaning aims at inducing sexual unions in the intellectual world in order to achieve certain effects in the physical world. I did everything with pleasure, made the conjuration he had taught me and believed with all confidence that I was now invisible. Immediately, I hurried to the Beth Hamidrash, the Jewish academy, went straight up to one of my comrades and gave him a good slap round his face. He, however, was no fool and returned the blow with interest. I was baffled and unable to understand how he could have recognized me since I had followed the instructions of the preacher with utmost accuracy [...].³⁹

The earliest known foundation of the Kabbalah is the *Sefer Yetzirah* (*Book of Creation*) whose origin and history is unknown, but which was in circulation at least since the 9th century. The book, a concise work consisting of a total of 65 paragraphs, is a proto-Kabbalistic work as it speaks, for example of the Sefirot, or attributes of God, but has not developed them yet into the system of the ten Sefirot which is central

³⁹Maimon, *ibid.*

to the Kabbalah. The creation as told in the book is a creation of the world through letters. The fourth chapter tells it as follows:

1. There were formed seven double letters, Beth, Gimel, Daleth, Kaph, Pe, Resh, Tau, each has two voices, either aspirated or softened.

[...]

3. These seven double letters He formed, designed, created, and combined into the Stars of the Universe, the days of the week, the orifices of perception in man; and from them he made seven heavens, and seven planets, all from nothingness [...].⁴⁰

In the next paragraph, these letters create things by the virtue of an algorithm:

4. From two letters, or forms He composed two dwellings; from three, six; from four, twenty-four; from five, one hundred and twenty; from six, seven hundred and twenty; from seven, five thousand and forty; and from thence their numbers increase in a manner beyond counting; and are incomprehensible.⁴¹

What is being described here is precisely the combinatory, mathematical law of permutation. According to this law, two discrete elements can be permuted—or shuffled— $2! = 2 * 1 = 2$ times, three discrete elements $3! = 3 * 2 * 1 = 6$ and seven elements $7! = 7 * 6 * 5 * 4 * 3 * 2 * 1 = 5040$ times. It is the same mathematical law that is at the heart of dodecaphonic and serial music composition, anagram poetry and computed word permutation poetry like that of Brion Gysin (whose *IN THE BEGINNING WAS THE WORD* permutes six words 720 times according to the same principle described in the quotation above). In the divine creation according to the *Sefer Yetzirah*, letter combinations function as a straightforward algorithmic source code. Unlike the executable code of a magical spell with its metaphoric and metonymic qualities, the execution is strictly formal, a proper computation. The second chapter of the *Sefer Yetzirah* even describes a mechanical computing device:

1. The foundations are the twenty-two letters, three mothers, seven double, and twelve single letters.

[...]

⁴⁰*Sepher Yetzirah or The Book of Creation*, translated by W.W. Wescott (1887), <http://www.sacred-texts.com/jud/yetzirah.htm>

⁴¹*Sepe Yetzirah*, *ibid.*

4. These twenty-two letters, the foundations, He arranged as on a sphere, with two hundred and thirty-one modes of entrance. If the sphere be rotated forward, good is implied, if in a retrograde manner evil is intended.⁴²

The second paragraph describes another combinatory algorithm, namely that of combinations—calling them “modes of entrance”—of two elements according to the formula $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19 + 20 + 21 = 231$. The “sphere” may therefore be imagined as a device with two mobile, concentric circles each of which have the 22 letters inscribed. The device can be used to compute all possible 231 letter pairs, on the premise that, unlike in a permutation, a combination like “aleph+bet” is the same as “bet+aleph.” This assumption still applies in the modern mathematical definition of a “combination.” The letter “aleph” can be combined 21 times with the remaining 21 letters of the alphabet, the next letter “bet” only 20 times since the combination with “aleph” had already been exhausted, and so on, resulting in the above formula. The creation of the world is, according to this speculation, computational. Later Kabbalist Torah readings employed algorithmic methods for reconstructing or, in modern computer programmer terminology, reverse-engineering divine creation through and within the letter. The Torah was read, among others, as an acrostic (notaricon), as letter permutation (temurah) or a numerological code (gematria) for the name JAHWE. The ecstatic Kabbalah, practiced in 13th century Spain by Abraham Abulafia and others, become probably the first comprehensive speculative science and art of language computation.

Superficially, the model of creation through mathematical combinatorics in the *Sefer Yetzirah* appears to converge with Einstein’s belief that “God does not play dice with the universe.”⁴³ However, it imagines God to have computed it since a die is a simple, one-purpose computer. The difference lies alone in the respective algorithms, the stochastic computation of the die versus the combinatory computation of the concentric circles in the *Sefer Yetzirah*. Both models converge, with order turning into chaos and vice versa, in the imagination of Thomas Pynchon’s novels. In “Gravity’s Rainbow,” set in the historical context of Alan Turing’s computer research in World War II England,

⁴²*Sefer Yetzirah*, *ibid.*

⁴³Stephen Hawking discusses this claim in depth in his lecture *Does God Play Dice*, <http://www.hawking.org.uk/lectures/dice.html>

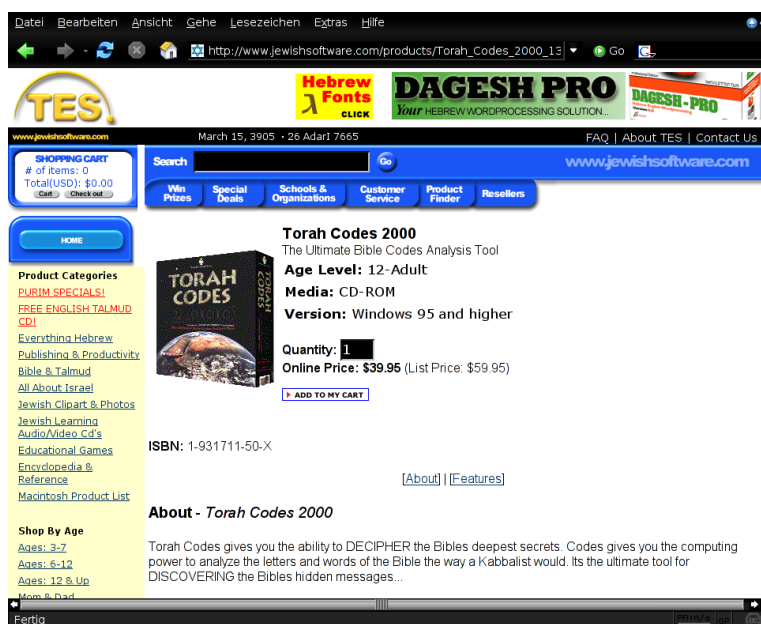


FIGURE 4. Advertising for Kabbalistic software

occult and computational methods are being used alike to predict the impact points of German missiles. The analysts include “Kabbalists who study the Rocket as Torah, letter by letter rivets, burner cup and brass rose, its text is theirs to permute and combine into new revelations, always unfolding.”⁴⁴ In Pynchon’s *The Crying of Lot 49* from 1967, an exploding spray can flies through a bathroom so that only “something fast enough, God or a digital machine, might have computed in advance the complex web of its travel.”⁴⁵ In Umberto Eco’s novel “Foucault’s Pendulum,” a computer named “Abulafia” is used to calculate Kabbalistic letter permutations of the name YHWE. The novel contains a BASIC source code of the program, commenting it with a detailed discussion of the combinatorics in the fourth chapter of the *Sefer Yetzirah*.⁴⁶ Kabbalah computer programs, as tools of the theoretical Kabbalah, however exist not only in fiction. The commercial PC software package *Torah Codes 2000* (see figure 4), available from Internet shops like <http://www.jewishsoftware.com> and Kabbalah

⁴⁴Thomas Pynchon. *Gravity’s Rainbow*. Vintage, London, 1995 (1973). [79], p. 717

⁴⁵Thomas Pynchon. *The Crying of Lot 49*. Perennial Classics, New York, 1999 (1967). [80]

⁴⁶Umberto Eco. *Foucault’s Pendulum*. Ballantine Books, 1990. [31]

Software <http://www.kabsoft.com>, is being advertised as the “ultimate Bible codes analysis tool.” The program renders the Torah as a database and features, among others, “Gematria look up of Word, Phrase, and sentence. Letter Substitution. Letter Analysis. Verse, Word, and Letter count. Bible statistics Query, and Search.”

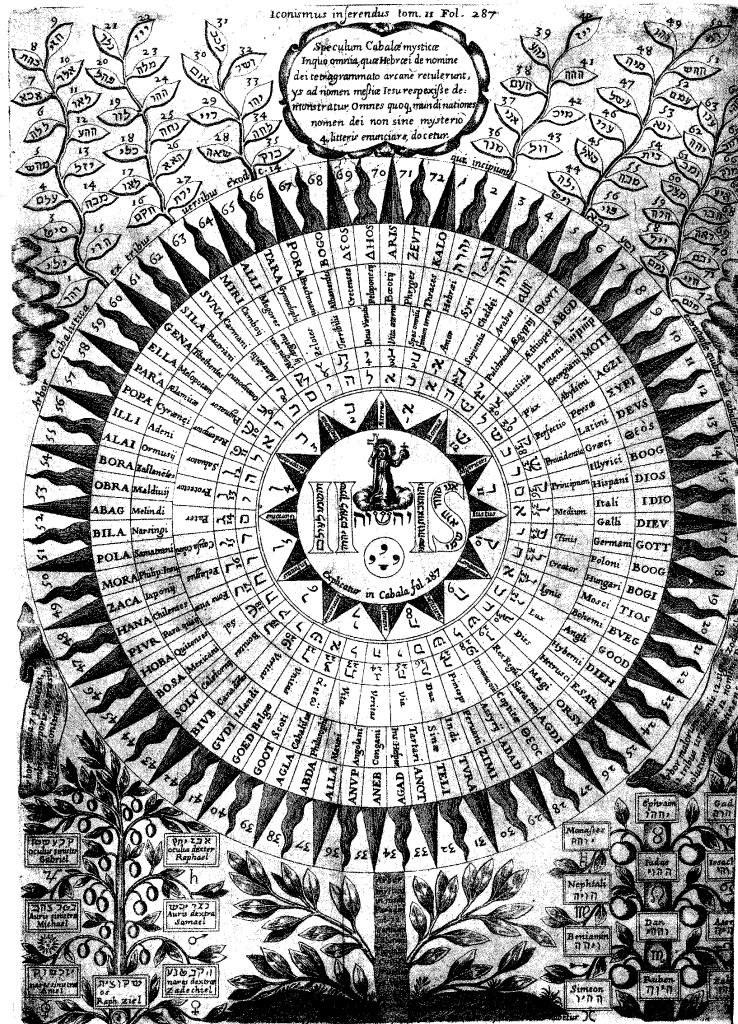


FIGURE 5. Diagram of the names of God in Athanasius Kircher's *Oedipus Aegyptiacus*

The idea of divine creation through computations of the letters of God's name was adapted by Christian Kabbalists in the Renaissance. In 1652-54, Jesuit priest and speculative scientist Athanasius Kircher published a book *Oedipus Aegypticius* that sought to reconstruct "Egyptian wisdom, Phoenician theology, Chaldaic astrology, Hebraic Kabbalah, Persian magic, Pythagorean mathematics, Greek theosophy, mythology, Arabic Alchemy, Latin philology."⁴⁷ The link between magic, Pythagorean thinking and Kabbalah is however being made here less through a historical analysis, but through syncretist combination. The book was founded on the assumption that all these practices and fields of knowledge contained residues of original Egyptian sciences and hieroglyphs.⁴⁸ Before Napoleon's discovery of the Rosetta stone, Hieroglyphs had not been deciphered in the modern age and became subject to occult-scientific speculation. Kircher's chapter on the Kabbalah contains a diagram of the ten Sefirot and a combinatory dial in the shape of a sunflower in which the tetragrammaton JHWH is permuted into 72 names of God (figure 5). On the outer concentric circles, these names get equated to four-letter names of god in contemporary European languages.

Kircher's model is obviously the Renaissance Italian Neoplatonist philosopher and first self-acclaimed Christian Kabbalist Giovanni Pico della Mirandola. Pico was first to make the Kabbalah known to a non-Jewish audience. He defended his Kabbalah studies against allegations of heresy in his *Oration on the Dignity of Man*.⁴⁹ In his writings, he christianized Jewish mysticism by pointing out trinitarian structures in the Kabbalah and extending the tetragrammaton with the Hebrew letter *Shin* so that it turned into "YHSVH," Jesus. The same occurs in Kircher's sunflower in which Shin is the central letter. Still, Kircher maintains the idea of creation through the divine letters and their permutation. After all, this is in accordance to the (Christian) Gospel of John and its claim that in the beginning there was the word. The things created and symbolized in Kircher's diagram are the seven planets and angels, the twelve signs of the Zodiac and tribes of Israel.⁵⁰

⁴⁷According to Joscelyn Godwin. *Athanasius Kircher*. Edition Weber, Berlin, 1994 (1979). [37], p. 57

⁴⁸ibid.

⁴⁹Giovanni Pico della Mirandola. *Oration on the Dignity of Man*. MacMillan, 1985 (1486). [76]

⁵⁰According to Godwin [37], p. 63

Ramon Llull and Lullism

A later work of Kircher, the *Ars magna sciendi sive combinatoria* from 1669, contains a permutation table very much like that in the *Sefer Yetzirah*. It lists the permutations of all integer numbers from 1 to 50 in a purely numerical, formal way. This book, however, is not an explicitly Kabbalistic work, but a recapitulation and partial modification of another complex of speculative language computation, the *Ars* of 14th century Catalan monk Ramon Llull, or Raimundus Lullus.

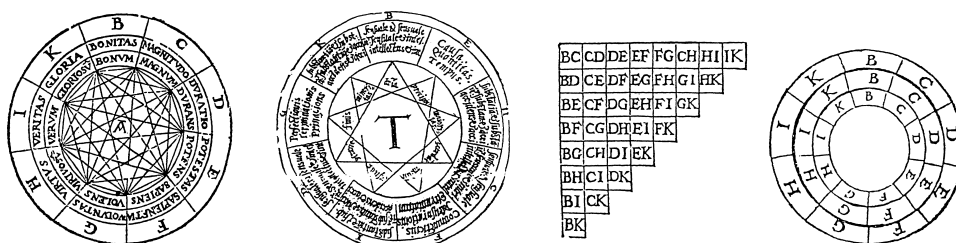


FIGURE 6. The four algorithms of Llull's *Ars*

The *Ars* is a shorthand for a formal-computational system of composing and deriving philosophical-theological statements Llull laid out in two books, *Ars generalis ultima* (1305) and, in shorter version, *Ars brevis*. The roots of the Llull's ars lie in a mystical revelation in 1265 on mount Randa on the island of Mallorca. During this event God allegedly revealed his own attributes to Llull. In the ars, these nine attributes are systematized and indexed with letters from B-K as follows: B – *bonitas* (goodness), C – *magnitudo* (magnitude), D – *duratio* (duration), E – *potestas* (power), F – *sapientia* (wisdom), G – *voluntas* (will), H – *virtus* (virtue), I – *veritas* (truth) and K – *gloria* (glory). Llull's nine divine attributes bear striking resemblance to the ten divine attributes of the Sefirot: 1. *Keter*, crown, 2. *Hokmah*, wisdom, 3. *Binah*, intelligence, 4. *Hesed*, love, 5. *Ge-vurah*, power, 6. *Tifaret*, compassion, 7. *Netzah*, endurance, 8. *Hod*, majesty, 9. *Yesod*, foundation, 10. *Malkut*, kingdom. It has been assumed, among others by Kabbalah scholar Moshe Idel, that Llull took his inspiration less from God himself than from 13th century Spanish ecstatic Kabbalah.⁵¹ Only one century after Llull, Pico della Mirandola describes what he calls the “ars raimundi” as a second form of

⁵¹Moshe Idel. Ramon Lull and Ecstatic Kabbalah. *Journal of the Warburg and Courtauld Institutes*, 51:170–174, 1988. [49]

Kabbalah. The theological idea behind Lull's system is that the nine attributes should be universally valid across all cultures and religions, so that the *Ars*, by providing an objective, formal system for creating statements from these universal truths, could prove the single truth of Christian religion and be used as a missionary device. Lull, as a matter of fact, did make several mission travels to Muslim countries in his lifetime. According to legend, he died a martyr in 1316, having been stoned to death by Muslims.

	FIG. A	FIG. T	QUESTIONS	SUBJECTS	VIRTUES	VICES
B	goodness	difference	whether?	God	justice	avarice
C	greatness	concordance	what?	angel	prudence	gluttony
D	eternity	contrariety	of what?	heaven	fortitude	lust
E	power	beginning	why?	man	temperance	pride
F	wisdom	middle	how much?	imaginative	faith	accidie
G	will	end	of what kind?	sensitive	hope	envy
H	virtue	majority	when?	vegetative	charity	ire
I	truth	equality	where?	elementative	patience	lying
K	glory	minority	how?	instrumentative	pity	inconstancy

The nine attributes form the nucleus of a table which juxtaposes them with nine logical relations (like *is equal to*, *is smaller than*, *is greater than*), nine questions as they were taught in rhetoric as part of the *inventio*, i.e. the gathering of ideas—"how," "what," "where from," "by what" etc.—, nine cosmological entities from God to angels, heaven, man, etc. down to instruments, in analogy to the Neoplatonist ordering of the cosmos as a hierarchy of hypostases, or instances, in which each instance is a minor reflection of a higher instance. Finally the Christian virtues and vices form two categories, and for this purpose are expanded from seven to each nine in order to fit the system.

The resulting "tabula" is what computer science calls a "flat" database with index fields (B-K). Through the unified number of nine entries per category, the index can reference any column. It provides, as a symbolic code, an abstract "alphabetum," or artificial meta-language for the entries of the table. The letter "A" is omitted in this artificial language, expressing the taboo of representing god as the absolute beginning and therefore the first letter in the alphabet. Lull's letter B-K may be the first example of what computer science calls the "semantics" of a programming language. From a linguistic standpoint, it is odd to call a language semantic, meaningful, which expresses no meaning in the sense of a judgement or interpretation, or reference to an idea. Instead, formal languages describe purely technical manipulations of symbols that require no cognitive interpretation, like "substitute all occurrences of the letter a with the letter b" as opposed to "substitute the melancholic tone of a text with an optimistic tone."

The second operation would require (significantly advanced) artificial intelligence, and it has not yet been proven that it can be achieved through a more complex formal manipulation of symbols in a satisfactory way (see p. 106). The “semantics” of a programming language however have nothing to do with artificial intelligence or cognitive computing, but simply refer to the style in which the artificial language is coded, i.e. which denominators, metaphors and other semantic handles are being used to reference non-semantic operations. Llull’s “alphabetum” demonstrates that no formal code functions without such a semantics, and culture coded into it. Any such code is thus a user interface—Llull makes the Latin alphabet the user interface of his system—and any code, whether assembly language or an iconic computer desktop, inevitably is an anthropomorphism that involves translation of processes into human-readable signs.

Llull’s formal alphabet however is not yet a programming language. As the index of a tabular, “flat” database, it references only data, not algorithms. His *Ars* includes four algorithms for transforming elements of the table into statements, but they are non-alphabetically represented by four circular graphic diagrams or “figurae” (see figure 6). The first *figura* links every of nine “principia absoluta,” i.e. god-given attributes, to every other of those principles in the grammatical form of an attributes. The combination “BC” results in the statement “goodness is great,” “BD” in “goodness is constant,” etc. The figure works like an exhaustive cross-reference, or hypertextual linking of all possible combinations, in this case $9 * 8 = 72$ if one excludes tautological combinations like BB, “goodness is good.”

The second figure links every logical relation to one of three subject matters. It therefore has a similar function as the cross-reference in figure one.

























The third figure yields all possible two letter combinations. In that, it differs from the first *figura* because it considers, just as in the modern mathematical understanding of a combination, “BC” the same as “CB” and results in $8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 = 36$ letter pairs. This is exactly the same combinatory algorithm as the one used earlier in the *Sefer Yetzirah* to determine 231 names of god from pairing 22 letters.

The fourth figure is used to combine three letter combinations instead of just pairs. Again, there is a resemblance to the *Sefer Yetzirah* and the device with concentric, rotating circles described in its second chapter. Llull writes that “in hac quarte figura, et ultima comprehenditur Ars tota,” that his whole *Ars* is contained in this fourth

figure. Yet it can't be used without restraints. Redundant combinations like "BBB" are, again, excluded. Through a complicated, and mathematically incoherent procedure which would take up to much space to explain here, Lull arrives at $28 * 9 = 252$ three letter combinations which may be generated with the fourth figure. Unlike the *Sefer Yetzirah*, Lull's ars does not conceive of letter permutations, but of combinations only, and only renders combinations of two elements mathematically correct.

Aside from these formal limitations, there are implicit semantic limitations and self-restraints in Lull's *Ars*. For example, a reading of combination BDC as "Deus—contrarietas—magnitudo," God is contrary to magnitude, would be invalid, respectively illegal. A whole set of heretical applications of the "ars" needs to be suppressed: which Lull's writing, however, does not mention. The system only allows building statements which are "true" a priori and then, applying its transformation methods, derive different true statements from them through formal-algorithmic means. This make Lull's ars a first prototype of modern symbolic logic in which logical statements are transformed according to purely formal rules. This resemblance is not coincidental. Leibniz, the inventor of symbolic logic, took his inspiration from Lull's *Ars* when he wrote his first book, *Dissertatio de arte combinatoria* in 1666, 12 years after Kircher's *Ars magna sciendi*.

After Giordano Bruno's theosophical Lullism in the 16th century, Lull's *Ars* was resurrected in the early 17th century through the encyclopedist Johann Heinrich Alsted. Kircher's *Ars magna sciendi* later marked the end point of 17th century scientific Lullism. Via Alsted, Lullist combinatorics changed its character in the 17th century from a theological device to a generative classificatory system of knowledge. Before Diderot and d'Alembert and their revolutionary reinvention of the encyclopedia in the late 18th century, knowledge in encyclopedias was not structured arbitrarily by the alphabet, but in a systematic order of things according to their place in a cosmology. Lullist combinatorics allowed the generation of complex hierarchical systems for knowledge classification through the exhaustive combination of categories. While Lull's use of algorithms was synthetic and meant to create complexity, a wealth of statements and reflections from only nine "absolute principles," the encyclopedist appropriation of his method should on the contrary handle and reduce complexity, structuring a given, unordered, large body of knowledge. It was probably the first historical example of outsourcing organization to algorithms, not unlike computer algorithms that manage payments and bank accounts today.

70	Orbis pictus							
4		Cornix cornicatur die Krähe krächzet	} á á	A a		Felis clamat die Katz mauzet	} nau nau	N n ⁵
		Agnus halat das Schaf blöket	} bé é é	B b		Auriga clamat der Fuhrmann ruft	} ó ó ó	O o
		Cicada stridet der Heuschreck sitztseht	} cí cí	C c		Pullus pípit das Kúchlein piepet	} pí pí	P p
		Upupa dieit der Wiedhopf ruft	} du du	D d		Cuculus cuculat der Kuckuck kucket	} kuk ku	Q q
		Infans ejulat das Kind wemmert	} é é é	E e		Canis ringit der Hund marret	} err	R r
		Ventus flat der Wind wehet	} fi fi	F f		Serpens sibilat die Schlang zischet	} sí	S s
		Anser gingrit die Gans gackert	} ga ga	G g		Graculus clamat der Häher schreiet	} tae tae	T t
		Os halat der Mund hauchet	} háh háh	H h		Bubo ululat die Eule uluhet	} ú ú	U u
		Mus mintrit die Maus pípfert	} í í í	I i		Lepus vagit der Hase quäket	} vá	W w
		Anas tetrínit die Ente schnackert	} kha kha	K k		Rana coaxat der Frosch quakert	} coax	X x
		Lupus ululat der Wolf heulet	} lu ulu	L l		Asinus rudit der Esel iahet	} yy y	Y y
		Ursus murmurat der Bär brummet	} mum mum	M m		Tabanus dieit die Breme summet	} ds ds	Z z

12 die Schlang | die Schlange A, B, C
22 iahet, gedwicks ygaet A, B, C

FIGURE 7. Comenius' *Orbis pictus*: A graphical interface for alphanumeric code

Still in the 17th century, the theologian, hermetic philosopher and educational reformer Jan Amos Comenius developed Lullist computational encyclopedism into a knowledge system with a graphical user interface (figure 7). A pupil of Alsted, Comenius wrote an immensely popular encyclopedist work with his schoolbook *Orbis pictus*.⁵² The *Orbis pictus*, *the world in pictures* was the first illustrated children's book in history. Each double page shows one aspect of the world—the planets in the beginning, then one area of human civilization like, for example, agriculture—, marking up every illustration with numbered footnotes that explained the objects both in Latin and the pupil's native language. With the book, pupils would learn both the order of the world and their native and a foreign language. Comenius' idea, inspired by the utopian thinkers Campanella and Johann Valentin Andreae (see p. 51 and 52), was to use pictorial language as firstly a universal language and secondly as an aesthetically concrete means to represent an abstract order of knowledge. Just like the graphical user

⁵²Jan Amos Komenský. *Orbis sensualium pictus*. In *Opera Omnia*, volume 17, pages 69–271. Academia Praha, Praha, 1970. [57]

interface of the early Macintosh competed with the alphabetic command line interface of DOS and Unix, the pictures in Comenius' book compete with Lull's nine letter alphabet as the interface, or "semantics," of a formal system.

Although there were radical confluences of Lullism and Christian-kabbalistic mysticism later in the 17th century (see p. 46), Lull and the 17th century scientific Lullists do not, unlike the Jewish kabbalists, conceive of letter combinatorics as the source of creation. Instead, they treat it as method of logical reasoning, generation and classification of statements and knowledge. God is implicit, not explicit in their combinatorial systems. The systems do not serve a kabbalistic reconstruction or reverse-engineering of a divine language. The divine is present as an inscription and representation of divine order within the categories, but God does not materialize in the computations. Performing them is not a theurgic act, as opposed to Kabbalism and, for example, Rabbi Loew's creation of the golem through practical Kabbalistic application of the *Sefer Yetzirah*.⁵³ The secularization of the Jewish Kabbalah through Lull and Christian Kabbalism continues with 17th century Lullist science, analytic philosophy since Leibniz, up to the 20th century concept of machine computation and computer software. It leaves, on the other hand, the question as to what extent religion, metaphysics and speculative thinking might still be present in contemporary computer culture.

Rhetoric and poetics

Classical rhetoric. Next to magical, Pythagorean, and Kabbalist thinking, an independent tradition of poetic language computations exists in classical rhetoric and poetics. A 1874 monograph of German poet and oriental scholar Friedrich Rückert, *Grammatik, Poetik und Rhetorik der Perser (Grammar, Poetics and Rhetoric of the Persians)* suggests, with its examples of word permutations and rotary dial word combinatorics in ancient Persian rhetoric and poetry, that rhetorical language computation might have non-Western sources.⁵⁴ Other prominent examples of computational writing are the Chinese *I Ching* oracle (see p. 77) and Tibetan prayer wheels. In fact, no historical Western computational text has been transcribed as early

⁵³The legend of Rabbi Loew has been told many times in books and films, see http://www.pantheon.org/articles/r/rabbi_loeb.html

⁵⁴Friedrich Rückert. *Grammatik, Poetik und Rhetorik der Perser*. Verlagsbuchhandlung Otto Zelle, Antiquariat Otto Harrassowitz, Wiesbaden, Osnabrück (Gotha), 1966 (1874). [86]

and frequently into electronic computer software as the *I Ching* (see p. 77). Classical Greek and Roman rhetoric advocated the virtue of “copia,” multitude of expressions and wealth of variation in speech. Part of this virtue was the ability to create an abundance of speech from a limited amount of ideas and material. The part of rhetoric that taught brainstorming for an oration, the “inventio,” was concerned with this problem. Word permutation became one of generative means of brainstorming and text composition, for example in a poem of the Greek poet Athenaios of the second century B.C.. It had its closest parallel in an elaborated form of chiasm—i.e. the crossing of two similar phrases—called “commutatio” or “permutatio,” like, for example “ego tu sum, tu es ego” (“I am you, and you are me”) in line 721 of Plautus’ *Stichus*, a comedy that made fun of Plato’s *Symposium*.



FIGURE 8. Sun’s *Looking Glass* computer desktop: spatial-pictorial representation of items very similar to the *ars memoria*

In classical rhetoric, “inventio,” the creation of topics, dialectically corresponded to “memoria,” the mental recollection of those topics in the act of speech. Memoria taught the memorization of a speech in an age where paper notes were unaffordable. The mnemotechnics of classical rhetoric is described in the Latin *Rhetorica ad Herennium*

which attributes its origins to the Greek poet Simonides.⁵⁵ It is a very particular, today obscure, system of visual-metaphoric memorization based on the imagination of architectural spaces, typically houses with separate rooms. These rooms are mentally filled with objects that represent concrete and abstract topics of the speech, by the virtue of metaphor, onomatopoetic resemblance or other figures of speech. While *memoria* involves no algorithmic computations in itself, it could be regarded as the first implementation of a visual user interface for alphanumeric codes, and certainly influenced Comenius' *Orbis pictus*, too. The "memory palaces" and "memory theaters" developed later in the Renaissance bear a striking resemblance to GUI (graphical user interface) desktops and their representation of files through icons arranged in a space. Rhetorical *memoria* even anticipates three-dimensional user interfaces (like *3dwm* or Sun's *Project Looking Glass*) in practically every aspect of the design.

Visual mnemotechnics became obsolete as paper became an affordable and more capable medium for scrap notes, a development that might have also have implications for software user interfaces. The question remains whether extending the visual illusionism and immersion of computer user interfaces to 3D simulation will really yield systems superior to software interfaces based on the notation and grammar of written language. If one compares graphical computer interfaces in their first mass market incarnation, the Apple Macintosh desktop of 1984, to contemporary graphical user interfaces on MacOS, Windows and the X11 Window System, it is obvious that much of the original iconic representations—the desk, the waste paper basket, the file cabinet—became obsolete and that the visual metaphors took up a life of their own. As opposed to a Macintosh of 1984, it is hardly possible to introduce a novice computer user to a contemporary graphical user interface by pointing out its resemblance to an analog desktop. With MacOS X, the original mass market graphical operating system in fact reverted to also providing a Unix command line shell.

Rhetoric as such can be seen as a discipline concerned with the formalization and quasi-formal manipulation of language. It is not genuinely formal because its tropes, such as metaphors, are semantic and not just the purely syntactical operations that would be necessary in a computation. Still, the concept underlying rhetoric is that any speech and writing can be created through formal instruction,

⁵⁵Frances Yates. *The Art of Memory*. Routledge & Kegan Paul, London, 1965. [101]

and—unlike in concepts of divine inspiration or artistic genius (see p. 106)—composition is a technique that can be learned by anyone. The more formal rhetorical instruction became, the more it approached language computation. In the step from chiasm to word permutation, poetic formalism finally went from semantic to formal-mathematical manipulation of symbols. Such texts have input data (the words to be transformed), an algorithm (permutation, often explained separately, just as in an algorithmic source code) and an execution, for example in writing down the multiple output of a permutation.

Proteic poetry. In the fourth century A.D. permutational poem *Carmen XXV* written by the Latin poet Optatianus Porfyrius, twenty stanzas are generated from a single stanza of source material.⁵⁶ These source words can be permuted within and between single lines. According to the mathematical law of permutation, more than 1.6 billion permutations of the stanza exist. The poem ironically relates the semantics of the words to the syntax of their computation since it tells about permutations and confusion in the songs of the muses. Another early known example of word permutational verse is a poem of the medieval Irish monk Dicuil.⁵⁷

In 1561, humanist scholar Julius Caesar Scaliger for the first time described word permutational poetry as a regular poetic form. In his *Poetices*, a poetics that become canonical for 16th and 17th century poetry, the form is called “Proteus” after the god who perpetually changes his face.⁵⁸ It includes a proteic poem “Perfide sperasti divos te fallere Proteu” (“Perfidiously, you, Proteus, hoped to fool the Gods”) whose words can be permuted at will as long as the hexametric meter is kept. As a humanist, Scaliger conceived of his *Poetices* as a continuation of the Greek and Latin rhetorical and poetic tradition. Most probably, his “Proteus” verse had Optatianus Porfyrius’ *Carmen XXV* as its model. Through Scaliger’s canonization and sanctioning of the form in the Renaissance, a boom of word permutational poetry resulted, especially in the early 17th century. The form was popular particularly among Christian ecclesiastical poets writing in Latin. In

⁵⁶Publilius Optatianus Porfyrius. *Publili Optatiani Porfyrii Carmina*. Paravia, Turin, 1973. [77]

⁵⁷Anagrams likewise are permutations of letters; however, since anagram poems never or rarely ever permit all possible letter permutations, there formal-mathematical permutations typically do not correspond to the semantically permitted permutations.

⁵⁸Julius Caesar Scaliger. *Poetices libri septem*. Frommann, Stuttgart, 1964 (1561). [89]

the period of the Thirty Years War, with its destruction of large parts of Middle Europe, proteic peace prayers were written whose permutations often span, like a hardcopy of computer program output, several dozens or hundreds of book pages. In them, language computation had become theurgy again, the activation of divine powers in the celestial macrocosm to change matters in the earthly microcosm.

Since the permutation of Scaliger's verse "Perfide sperasti divos te fallere Proteu" was restrained through the hexameter, not mapping its mathematical permutation, the proteic verse form became modified later in the 17th century in attempts to combine it with Lullist combinatorics. Thomas Lansius, a professor of rhetoric and politics, wrote a proteic poem whose two lines consisted only of monosyllabic words. They could be shuffled without metrical constraints. In fact, his poem merged the "Proteus" with a monosyllabic double verse Scaliger calls "Correlativi" (and which are known as "versus rapportati" otherwise).⁵⁹ Through the permutation of monosyllabic words, the poetic combinatorics of the verse for the first time was identical to its mathematical combinatorics, and the rhetorical and poetic tradition of Athenaios, Optatianus and Scaliger for the first time converged with Lullism.

Johann Heinrich Alsted consequently reprints Lansius' poem along with a computation table of its permutations in his Lullist encyclopedia.⁶⁰ Leibniz' *Dissertatio de arte combinatoria* is, through its many examples and citations, one of the richest sources of 17th century proteic poetry.⁶¹ Between Alsted and Leibniz, the mid-17th century German poet, language researcher and Lullist Georg Philipp Harsdörffer sought to systematically exhaust the potential of combinatorics in linguistics and poetics. Not only did he write two Proteic monosyllabic poems after Lansius' model—and with a footnote that credits Lansius—, but he also designed a combinatory morphologic word creation machine after the model of Lull's third "figura." The *Fünffacher Denckring der teutschen Sprache (Five-fold Thought Ring of the German Language)* (figure 9) should permit anyone to generate

⁵⁹Scaliger, *ibid.*

⁶⁰Johann Heinrich Alsted. *Encyclopaedia*. Holzboog, Stuttgart (Herborn), 1989 (1630). [4]

⁶¹G.W. Leibniz. *Dissertatio de arte combinatoria*. In *Sämtliche Schriften*, volume 1 of VI, pages 165–230. Akademie-Verlag, Berlin, 1989. [62]



FIGURE 9. Georg Philipp Harsdörffer's *Denckring*, a word generator

all existing and potential words of the German language by the combination of what Harsdörffer calls syllables, but which in modern linguistic terminology are morphemes. At this point, computation is no longer a rhetorical, magical or theurgic means of manipulating language. Instead, language in itself is thought to be computational and algorithmic, a program. While Lull, in his anticipation of symbolic logic, separates what is later called “artificial” from “natural” language with his index letter B-K versus the word entries of the “tabula,” Harsdörffer thinks of everyday natural language as being identical to artificial combinatory language. All language thus is programming language. This idea also sets Harsdörffer apart from the Kabbalists who assigned computational power to divine and theurgic language only—the code of creation—, not to everyday language.

An ultimate hypertrophy of everyday language into program code is reached with the sonnet XIV. *Libes-kuss: Vom Wechsel menschlicher Sachen* (14th Kiss of Love: On the Permutation of Human Matters) written in 1671 by the German poet, and later heretical “prophet” and

“monarch” Quirinus Kuhlmann.⁶² Adapted straightforwardly from Harsdörffer’s second proteic poem, reusing most of its words and word poems just like an open source program reuses earlier source code, it also mixes language material from Solomon’s proverbs and, very likely, the preamble of Johann Valentin Andreae’s Rosicrucian *Fama* of 1614 (see p. 51). The result is a monumental proteic sonnet whose permutations even the author can no longer calculate. The afterword stops at spelling out the permutation of 50, a number copied straightforwardly from the permutation table in Athanasius Kircher’s *Ars magna sciendi sive combinatoria*. (Kuhlmann uses Kircher’s modified Lullist letter code in other poems of the same volume.) The sonnet however is composed not of 50, but of $12 * 13 = 169$ words, amounting to a total of $13!^{12} = 3.399 * 10^{17}$ permutations. It functions as a world machine, permuting, both in the meaning of its words and in its combinatory mechanics, an inventory of the micro- and macrocosm. Its own couplet, and its afterword, claims that to grasp the principle of the permutation of things means to have wisdom of the world. This alludes to the both to the use of Solomon’s proverbs in the poem and adaptations of Solomon’s Song Celestial in the volume in which it appeared. Through this intertextuality, the poem renders itself a Solomonic machine. It is a computational reverse engineering of Solomon’s wisdom, considering the proverbs as they are written in the Bible the fragmentary output of an occult machine.

Beyond being the principle of wisdom, permutation also is the principle of micro- and macrocosm itself: “everything permutes, everything loves, everything seems to hate something,” as the first line of the couplet puts it. Not only does the poem enrich rhetoric and poetics with encyclopedic Lullism. It also practices Christian Kabbalah through its concept of creation as the permutation of words, in a similar manner as in the *Sefer Yetzirah*.

Combinatory poetry and the occult

Written in 1671, Kuhlmann’s poem concludes 16th and 17th century word permutation poetry, manifesting the climax and temporary end point of the form. By its hypertrophy of word permutations into a cosmology, it contradicts the secularist tendency of 17th century Lullism of reassessing an originally theological device as a rational method of knowledge classification or, as in Harsdörffer, linguistic analysis

⁶²Quirinus Kuhlmann. *Himmlische Libes-küsse*. Niemeyer, Tübingen, 1971 (1671). [60]

and synthesis. Leibniz who on the grounds of rational science transforms Lullism into symbolic logic and an early mechanical calculation device, stands for an opposite to Kuhlmann's Kabbalist cosmology.

In a less baroque and more minimal-modernist fashion, Brion Gysin's permutation poem *IN THE BEGINNING WAS THE WORD* manifests an epistemology with striking parallels to Kuhlmann's, regardless of its employment of a Honeywell digital computer as the end product of rationalist computation. Gysin conflates divine speech, creation through speech and language permutation as an ecstatic practice just like Abulafia and Kuhlmann did before him. While the poetic algorithm is the same and its metaphysical connotations are similar, the implications differ. The theosophic and gnostic tradition in Gysin's work is a "dark," satanic one. It could not be thought of without Aleister Crowley's satanic travesty of theosophy and Kabbalah (such as in *The Book of Lies*).⁶³ As in the work of William S. Burroughs, ecstasy is linked to psychedelic drugs and revelation of a Freudian unconscious rather than religious epiphany.

While Kuhlmann remained within the boundaries of Christian faith, he too ended up as a heretic. Two years after the sonnet was published, he embarked on an occultist-chialistic career. His book *Neubegeisterter Böhme (Newly Inspired Böhme)*, a work ostensibly on the German mystic Jakob Böhme) re-employed the Lullist combinatory method for generating one thousand "theosophical questions." From there on, Kuhlmann considers himself a prophet, travels through Europe to forge an alliance of Muslims, Lutherans, Orthodox and Calvinists against the pope, fails to convert the sultan in Constantinople, and ends up being burned as a heretic in Moscow.—In striking similarity, Abraham Abulafia tried to bring down the pope with his ecstatic Kabbalah four centuries before.⁶⁴—On his itinerary, Kuhlmann writes a monumental book *Kühlpsalter (Cool Psalm)*, his testament and spiritual poetic diary that in parts continues to employ combinatorial-permutational methods of text generation and word-play.

Both Abulafia's and Kuhlmann's biographies exemplify how computational execution of language transgresses the formal and intellectual realm, turning into a radical practice. It appears that the semantics of executable code does not only reside in its culturally chosen, arbitrary denominators (like Lull's letters B-K), but also in the very

⁶³Aleister Crowley. *The Book of Lies*. Red Wheel Weiser, 1970 (1913). [26]

⁶⁴Moshe Idel. *The Mystical Experience in Abraham Abulafia*. State University of New York Press, Albany, 1988. [48]

act of execution. Once formal execution is considered a cosmological principle, it becomes synonymous with performative execution. The same formal-performative extremism can be found in the more rigorous early 1960s Fluxus action scores like La Monte Young's instruction to "Draw a straight line and follow it."⁶⁵ It implies a philosophical defiance of space and time constraints, and reverses subject and object in that the artifice, the line, becomes the subject and the performer its follower, respectively object. In Burroughs' *The Electronic Revolution*, technology turns into an autonomous subject, too: language becomes a biological virus, tape recorders take upon the identity of Adam and Eve, typewriters turn into speaking bugs like in David Cronenberg's filmic adaption of *Naked Lunch*. The consequences are similar to those of Abulafia and Kuhlmann in that a poetics of executing code transgresses into performance and politics:

Here is a sample operation carried out against The Moka Bar at 29 Frith Street London W1 beginning on August 3, 1972 . . . Reverse Thursday . . . Reason for operation was outrageous and unprovoked discourtesy and poisoned cheese cake. . .

Now to close in on The Moka Bar. Record. Take pictures. Stand around outside. Let them see me. They are seething around in there. The horrible old proprietor, his frizzy haired wife and slack jawed son, the snarling counter man. I have them and they know it.

"You boys have a rep for making trouble. Well come on out and make some. Pull a camera breaking act and I'll call a Bobby. I gotta right to do what I like in the public street."

If it came to that I would explain to the policeman that I was taking street recordings and making a documentary of Soho. This was after all London's First Espresso Bar was it not? I was doing them a favor. They couldn't say what both of us knew without being ridiculous. . .

"He's not making any documentary. He's trying to blow up the coffee machine, start a fire in the kitchen, start fights in here, get us a citation from the Board of Health."

⁶⁵La Monte Young. Composition 1960 #10 to Bob Morris. In Harald Szeemann and Hans Sohm, editors, *happening & fluxus*. Kölnischer Kunstverein, Köln, 1970 (1960). [102]

Yes I had them and they knew it. I looked in at the old Prop and smiled as if he would like what I was doing. Playback would come later with more pictures. I took my time and strolled over to the Brewer Street Market where I recorded a three card Monte Game. Now you see it now you don't.

Playback was carried out a number of times with more pictures. Their business fell off. They kept shorter and shorter hours. October 30, 1972 The Moka Bar closed. The location was taken over by The Queens Snack Bar.⁶⁶

An extension of the logic of algorithmic code into political action exists, more prominently, in the Free Software movement founded by Richard Stallman with the *GNU Manifesto* in 1983.⁶⁷ The popular, anonymous hacker credo that “information wants to be free” supposes a political semantics embedded into formal, digital code, by its technical virtue of boundless and lossless replication. This replication already happens when program code gets executed, i.e. copied from a storage device into the CPU. The Free Software movement translates the logic of executable code into a number of other executable codes: the GNU manifesto as a political instruction code, the GNU licenses as a legal code, free software documentation as a technical instruction code. Even religious activation is involved, although ironically, in dubbing software experts and prominent hackers “gurus,” activists as “evangelists” and, finally, supreme guru and evangelist Stallman himself as *Saint Ignutius of the Church of Emacs*.⁶⁸

Unlike Kabbalism and its permutations, the computational poetic trope of the GNU project is recursion, the potentially infinite looping iteration of a statement (on which Douglas R. Hofstadter's book *Goedel Escher Bach* is one long meditation). “GNU” itself stands for “GNU is Not Unix.” It is, in other words, an acronym that contains itself, recursively expanding into “GNU is not Unix is not Unix,” ad infinitum. Recursion is a technical principle also of the Lisp programming language in which much of the Emacs is written. It is furthermore the core logic of the GNU General Public License (GPL) which

⁶⁶William S. Burroughs. *Electronic Revolution*. Expanded Media Edition, Bonn, 1982. [19]

⁶⁷Richard Stallman, *The GNU manifesto*, <http://www.gnu.org/gnu/manifesto.html>

⁶⁸Emacs is the universal text editor and integrated development environment of the GNU project.

commands that a derivative work of a GNU-licensed code—be it a modification or extension—must in turn be released under the terms of the GPL. Microsoft therefore calls the GPL a “virus” and refers to code licensed under its terms as a contagion.⁶⁹ Indeed, the larger the public body of GPLed program code is, the higher the incentive of programmers to use it for new projects. Since this requires the subsequent release of those projects under the GPL, the body of available GPLed code increases in turn and attracts even more developers to take from and contribute to it. “Freedom” is the metaphysical center of the movement. The GNU manifesto defines it only in its practical meaning for software and software usage, but not theoretically, in philosophical or political terms. It appears to be a freedom founded on the structure of executable code and digital information—an ontology derived from technical function. The place of a theory of freedom is taken by the figure of recursion. It becomes the poetic, philosophical and economic trope in which freedom both materializes and through which it can be grasped, described as in a source code and therefore, paradoxically, controlled.

Information as a code that executes into political action and into utopia existed before the Free Software movement. It was central to the 17th century educational utopias of Comenius, his collaborator and “Royal Society” founder Samuel Hartlib and their intellectual mentor and correspondent, German Protestant theologian and co-author of the original 1614 Rosicrucian manifesto *Fama Fraternitatis*, Johann Valentin Andreae.⁷⁰ Andreae’s 1619 pamphlet *Turris Babel* (*The Tower of Babel*) documents his passage from the Rosicrucian “ludibrium,” “playful fancy” as he called it later in his autobiography, to a more concrete educational reform politics.⁷¹ The text is a dialogical satire on the Rosicrucian craze Andreae himself had instigated and which, in the first five years only, had yielded more than 150 public replies from authors who sought to get in touch with the unknown hermetic brotherhood.

⁶⁹Microsoft states on its page <http://www.microsoft.com/resources/sharedsource/Articles/LicensingOverview.aspx> that the copyleft rule “is what makes the GPL ‘viral,’ because it causes GPL terms to apply to software that incorporates or is derived from code distributed under the GPL, regardless of whether the program’s developer intended that result or even knew of the presence of GPL code in the program.”

⁷⁰Johann Valentin Andreae. *Fama Fraternitatis, Confessio Fraternitatis, Chymische Hochzeit: Christiani Rosencreutz Anno 1459*. Calwer Verlag, Stuttgart, 1994 (1973). [6]

⁷¹Johann Valentin Andreae. *Turris Babel*. Zetzner, Strasbourg, 1619. [5]

With *Turris Babel*, Andreae joins that debate under his own name and mocks the Rosicrucian hype. However, instead of busting it and outing himself as the plotter, he brings up seventy-five allegorical protagonists who each pronounce their own opinion about the Rosicrucians. In chapter 16, three characters enter the scene, the “reformer,” the “deformator” and the “informer.” While the deformator wants to do away with all traditional ties and institutions including church and state, the reformator hopes for their restoration. The informer finally supersedes their debate by demanding to “inform” mankind so that “the divine law will be saved from the deformator’s corruption and the reformator’s eagerness and become the constitution of this world.”

“Information” refers to its literal Latin sense of “impregnation,” “shaping,” or “instruction.” The informer is an agent of a new *Christiana Societas* (*Christian Society*) which the final chapter of the book and Andreae’s later writings propagate and describe. With the Rosicrucians being superseded by the Christian Society, fama is superseded by information, respectively education. In the ideal state of this information society, Andreae’s utopian Christian-communist republic *Christianopolis*, all knowledge is denoted in public mural paintings. It is precisely the concept of the graphical user interface—borrowed from Tommaso Campanella’s 16th century utopian text *City of the Sun*—which Comenius implements in the *Orbis pictus*. Defined against de-formation, re-formation and fama, Andreae’s information is not only loaded with pedagogics and theology, its definition also is radically performative. Information is only what has an impact, reaching and impregnating its recipients just like the execution of a program code mobilizes matter. In Andreae’s *Christiana Societas*, the origin of the impregnation is “heaven,” the informant is called “God.” Almost four hundred years later, this analysis of “information” seems to be politically and philosophically more precise and rigorous than concepts like “information society” and “information wants to be free” which, passing off “information” as a culturally autonomous, ahistorical and self-perpetuating agent, in fact write a crypto-theology of information.

Setting information free through code and activism, and bringing down the Moka bar, both the Free Software movement and Burroughs’ *The Electronic Revolution* turn their theoretical Kabbalah into practical Kabbalah solely through their technical understanding of information, language, code. Executable code not only mobilizes matter, but also people. In Abulafia’s and Kuhlmann’s Kabbalism, the code and its execution takes up a life of its own. Both think of them not merely as

a reflection, but as a materialization of the divine, and develop this thought from theory into ecstatic and political-theological practice. The step from writing to action is no longer metaphorical, as it would be with a semantic text such as a political speech or a manifesto. It is concrete and physical because the very code is thought to materially contain its own activation; as permutations, recursions or viral infections. It is not only words made flesh, but words being flesh.

Computation as a figure of thought

Lullist imagination. Explicit and hidden theological politics seem to run through the various poetics and cultural practices of formally executable code, yet contradictory tendencies exist as well:

(1) Totalism vs. Fragmentation

(a) Totalism / Synthesis

The employment of algorithmics, permutation, combination and recursion to exhaust all existing aspects of a matter: The exhaustion of topics in rhetoric, the exhaustion of theological truth in Lull's *ars*, the exhaustion of knowledge in Lullist encyclopedism, the exhaustion of wisdom in Kuhlmann's poem through reverse-engineering Solomon.

Algorithmics functions as expansion of a small source code into a near-infinitude of material.

This tendency culminates in Quirinus Kuhlmann's theoretical sketch of an *Ars magna librum scribendi*, a universal letter combination machine designed to write all existing and potential books in the world (see p. 61).

(b) Fragmentation / Analysis

Generative classification of knowledge in turn compartmentalizes it into specialist domains. Program execution can be seen as the handling of complexity, the possibility of reducing, for example, one hundred pages of proteic verse permutations to one line of material and one paragraph of instruction; or the reduction of several thousands statements created by Lull to three algorithms and one table of terms.

In other words, combinatory knowledge classification embodies the dialectic that not only an abundance of information can be generated from a minimal source code, but that vice versa an abundance of information can be analytically reduced to one algorithm.

(2) Rationalization vs. Occultism

(a) Rationalism

The indexing and classification of knowledge, the development of artificial formal languages as “user interfaces” for data and algorithms, computation as such.

(b) Occultism

Theurgy, totalism expanding into demiurgic creation, code execution translating into practical political theology.

Kuhlmann’s example shows that theurgy and demiurgy are not necessarily precursors of a scientific rationalization, but that scientific rationalization (like that of 17th century encyclopedic Lullism) itself can be used as a philosophical ground for occult theology, precisely as an attempt to exceed scientific Lullism in a physical and metaphysical, microcosmic and macrocosmic grasp of totality. This further proves that the cultural history of executable code cannot simply be written as a linear Hegelian progress from magical to scientific practices.

(3) Hardware vs. Software

(a) Hardware

There is a tendency of building computational hardware for algorithms: the device described in the *Sefer Yetzirah*, Llull’s and Giordano Bruno’s combination wheels, Harsdörffer’s *Denckring*. Kuhlmann, too, designs a *Wechselrad* (*permutation wheel*) to speed up the permutation of his sonnet.

(b) Software

The rhetorical tradition of mental computation and memorization, in the “inventio” and “memoria.”

Abstraction from mechanical devices through symbolic handles and denominators, for example in the replacement of Llull’s mechanical “figurae” through abstract symbolic denominators in later symbolic logic and computer programming languages.

(4) Syntax vs. Semantics

(a) Syntax

Program code as purely formal and therefore syntactical abstraction from human language.

(b) Semantics

The inscription of metaphors and semantic handles, from Llull's letters B-K up to statements like "for," "while," "if" in programming languages.

(5) Artificial vs. Natural Language

(a) Artificial Language

The idea that language can be computed only through a purely formal, syntactical metalanguage that exists separately from natural language.

(b) Natural Language

The idea that common human language itself is a product of computation, and can be described with algorithms, such as in Harsdörffer's *Denckring*, Kuhlmann's poetry or artificial intelligence research.

CHAPTER 3

Computation as Fragmentation

Pre-modern concepts of computation typically relied on an equivalence of macrocosm and microcosm and so conceived of algorithms as demiurgic creation, metaphysically. The game by contrast is a central model of 20th century computation, both in the arts and in technology. The main ontological difference between theosophic and game computing is that games can do without a reference to higher powers—which is why religions declared many games, dice and card games for example, sinful—and may impose arbitrary restrictions that do not logically follow from a higher natural order. A game, in other words, can be its own autonomous, self-contained world.

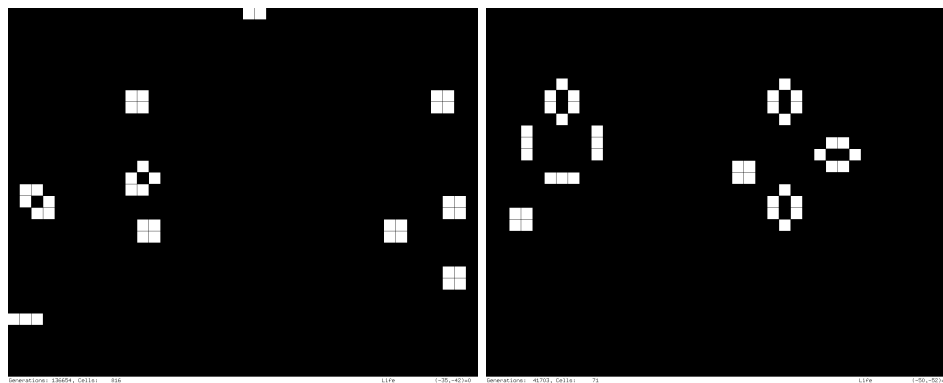


FIGURE 1. *xlife*, a computer program playing *Conway's Game of Life*

Any game is a process based on rules, a formal source code that can be expressed in logical language. There exist multiple models of computing as games. Best known might be *Conway's Game of Life* (figure 1) with its particular implementation of so-called cellular automata. Cellular automata consist of simple elements in a matrix that have a finite amount of states. Most typical is a binary state of either zero or one or, in a graphical matrix, black and white. Obeying

very simple transformation rules, the automata alter their states in dependency of the states of neighboring cells. Cellular automata have existed as a computational model since the 1940s. *Conway's Game of Life*, first published by the British mathematician John Horton Conway in 1970, is based only on the following rules: "If a black cell has 2 or 3 black neighbors, it stays black. If a white cell has 3 black neighbors, it becomes black. In all other cases, the cell becomes white."¹ Remarkably, this simple game allows Turing-complete computation. That means, any calculation can be made, and computer programs can be written on the basis of cellular automata. There also exist computer programming languages like *Logo* which are based on a game logic. Designed for children, *Logo* allows arbitrary computations with the help of the screen graphic of a turtle. Users have to program the turtle to make certain movements on the two-dimensional screen and this way perform calculations. In the 17th century, Georg Philipp Harsdörffer conceived of his linguistic and poetic combinatorics as games, worked under the name of "The Playing One" in a language research society and described most of his combinatory devices in dialogical fiction books that instructed poetry as a social game.² But the playful nature of these poetic and artistic language computations made them subject to parody and ridicule, later oblivion, from 1700 onwards until roughly 1900.

Gulliver's Travels

In 1705, German universal scientist Daniel Georg Morhof published a treatise *De Arguta Dictione*, which for the first time combined the Jesuit *acumen* rhetoric (see p. 22) of witty points with Lullism in a rhetoric based on combinatory principles.³ Next to Harsdörffer's mid-17th century works of scientific and poetic instruction, Morhof's book is one of the few systematic computational poetics, and theories of language and composition, based on algorithms. It stands, next to Quirinus Kuhlmann's poetry, as another culmination and end point of the 17th century boom of Lullist thought. In many respects,

¹Wikipedia concisely covers the subject in http://en.wikipedia.org/wiki/Cellular_automaton and http://en.wikipedia.org/wiki/Conway's_Game_of_Life.

²Georg Philipp Harsdörffer. *Mathematische und philosophische Erquickstunden*. Texte der frühen Neuzeit. Keip, Frankfurt (Nürnberg), 1990 (1636). [44] and Georg Philipp Harsdörffer. *Frauenzimmer Gesprächspiele*. Deutsche Neudrucke: Reihe Barock. Niemeyer, Tübingen, 1968-69 (1643-57). [43]

³Daniel Georg Morhof. *De Acuta Dictione*. Petrus Böckmannus, Lübeck, 1705. [70]

Lullism was a continuation of the medieval scholastic thinking in Aristotelian categories. Through Newton and scientific empiricism, and the shift towards individual genius and against poetic rules in literature, Lullism became outmoded. As an obsolete and seemingly bizarre scientific and poetic method it ended up being subject to rationalist parody.

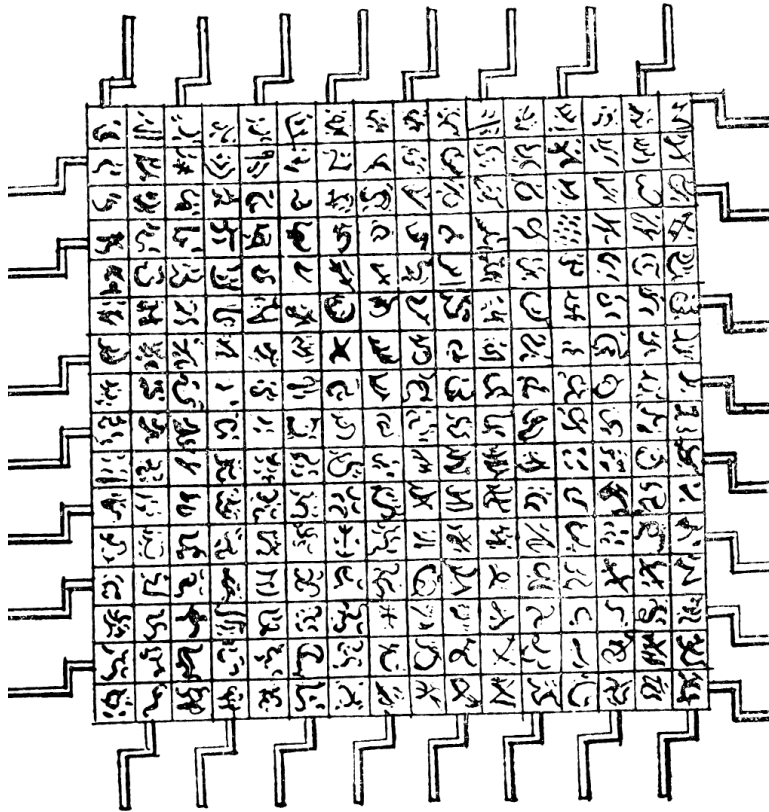


FIGURE 2. The writing machine in the Grand Academy of Lagado, illustration from *Gulliver's Travels*

In Swift's *Gulliver's Travels* from 1726, the first-person-narrator visits the flying academic island of Lagado and witnesses the mechanics of a combinatorial machine (figure 2):

The first professor I saw was in a very large room, with forty pupils about him. After salutation, observing me to look earnestly upon a frame, which took up the greatest part of both the length and breadth of the room, he said perhaps I might wonder to see him employed in

a project for improving speculative knowledge by practical and mechanical operations. But the world would soon be sensible of its usefulness, and he flattered himself that a more noble exalted thought never sprang in any other man's head. Everyone knew how laborious the usual method is of attaining to arts and sciences; whereas by his contrivance the most ignorant person at a reasonable charge, and with a little bodily labor, may write books in philosophy, poetry, politics, law, mathematics, and theology, without the least assistance from genius or study. He then led me to the frame, about the sides whereof all his pupils stood in ranks. It was twenty feet square, placed in the middle of the room. The superficies was composed of several bits of wood, about the bigness of a die, but some larger than others. They were all linked together by slender wires. These bits of wood were covered on every square with paper pasted on them, and on these papers were written all the words of their language, in their several moods, tenses, and declensions, but without any order. The professor then desired me to observe, for he was going to set his engine at work.⁴

Swift's "Grand Academy of Lagado" is believed to be a parody on the Royal Society, the British academy of sciences. The Royal Society was founded out of the *Invisible College* in the mid-17th century by, among others, Robert Boyle and astronomer and Lullist mathematician John Wilkins, and Johann Valentin Andreae's correspondent Samuel Hartlib. The *Hartlib papers* CD-ROM which documents the correspondence between Hartlib and others on the foundation of the Royal Society,⁵ shows that Lullist combinatorics was indeed a major subject of discussion and occupation in the *Invisible College*. Almost one century later, Swift writes a rationalist satire on what he perceives to be the speculative fancy of Lullism in academia. The chapter also mocks a universal language project in which words, i.e. abstract symbols, are replaced with concrete things, a parody, as it seems, on pictorial universal languages as they were envisioned by Campanella,

⁴Jonathan Swift. *Gulliver's Travels*. Washington Square Press, New York, 1960. [97], part 3, chapter 5

⁵Mark Greengrass, editor. *The Hartlib Papers*. University Microfilms, Michigan, 1996. CD-ROM. [41]

Andreae and Comenius (who was a close correspondent of the *Invisible College* and stayed in London between 1641 and 1642).

A shift in culture manifests in the fact that Swift's description of the text writing machine could work as a parody at all. In 1674, Quirinus Kuhlmann had envisioned, without a bit of irony, an *Ars magna librum scribendi*, a Lullist *ars* of writing books, which would mechanically generate all existing and all possible books.⁶ Unlike Swift whose fictitious machine computes foreign, unreadable letters, Kuhlmann thought, just as Harsdörffer, of human language as something inherently computable. therefore suffices as a potentiality and thought experiment on language and writing, and needs either an actual machine, nor its output to make its point.

The Library of Babel

The speculative premise of Jorge Luis Borges' short story *The Library of Babel* from 1941 is very similar to Kuhlmann's *ars magna librum scribendi*.⁷ It envisions a nearly total library whose books are generated from a combinatorial mechanism and arranged into a virtually infinite array of hexagons. Yet the story is told from the point of view of an inmate of the library. He has no access to its source code, but tells how this source code, or generative principle, became reverse-engineered. In the first sentence of the story, he refers to the library as "The universe (which others call the Library)," expressing a personal opinion that echoes cosmological Lullism. Like in other stories of Borges, the subjective account is a simulatenous device of uncertainty and irony. Fictitious editorial footnotes and remarks make the text a product of fake philology. Because of the subjective narrative perspective, all information about the library has to be taken with a grain of salt. The first-person narrator appears to be a librarian since he speaks of "the hexagons under my administration." But his reliability as an information source is no better than that of an inmate of Plato's cave. So it is merely his theory that "the Library includes *all* verbal structures, *all* variations permitted by the twenty-five orthographical symbols." In fact, the opposite can be concluded from his statement. He also says that "each book is of four hundred and ten pages; each page, of forty lines, each line, of some eighty letters which are black in color," so there are $80 * 40 * 410 = 1344000$

⁶In his Latin treatise Quirinus Kuhlmann. *Prodomus*. Lotho de Haes, Amsterdam, 1674. [59]

⁷Jorge Luis Borges. *The Library of Babel*. In *Ficciones*, pages 79–88. Grove Press, New York, 1941. [13]

characters per book, consequently $25^{1344000}$ books in total, then, from other data he gives, $32 \text{ books} * 5 \text{ shelves} * 5 \text{ walls} = 575 \text{ books per hexagon}$, and thus $\frac{25^{1344000}}{575}$ hexagons in the entire library.

But it is just the speculation of the first person narrator that the library can be explained by “combinative analysis,” as a computed whole. Twelve years before the story appeared, Gödel had found the logical paradox that formal systems cannot fully describe themselves, or, if they are consistent, cannot prove their own consistency. In the Library of Babel, the book that describes the computation and hence source code of the library, is only one arbitrary book of $25^{1344000}$ at least one of which contains a refutation of this theory. The cataloguing systems are part of the dilemma rather than its solution. There exist “thousands and thousands of false catalogues,” and a “catalogue of catalogues.” Through the letter combinatorics, differentiation between data (books) and metadata (catalogues) becomes arbitrary. The library ends up being a self-referential linguistic universe in which subject can no longer be told from object, and it is not clear what represents what. While the library may contain all knowledge—as envisioned by Kuhlmann earlier—it is incapable of classifying it. In computer science terms, the *namespace* of its classifications is either flat or infinite. And while Kuhlmann believes that the *Ars magna libri scribendi* expands knowledge and wisdom, the *Library of Babel* marks a melancholy and sceptical view of the same intellectual experiment: Through its totality, the library is contingent. Everything that can be thought has been thought in it already. At the same time, this melancholy subverts itself. After all, there is a story, written by Jorge Luis Borges, called *The Library of Babel*. This story is one clear-cut and solid block of reference and metadata. The Library of Babel might refute itself in any possible sense, but it does not refute itself being the *Library of Babel*. According to its inmate’s speculation, it is objective, mechanical, without history, existing “*ab aeterno*.” Yet the library functions only through its human inmates who read the books. Without them, it would be just storage of blotted paper. The books are meaningless without human interpretation. While all human acts might be anticipated in the book, those acts don’t exist without the acts of reading. The melancholic contingency of writing, it follows, is the motor of its interpretations. Only the interpretations, i.e. the cultural appropriation of what only appears to be a natural, not human-made artefact, make the library what it is. Its code is cultural even if no culture, subjectivity and interpretation were involved in its initial creation.

Finally, the account of the first person narrator has to be seen as a product of the library. So it implies its own refutation—a refutation that such a library does exist at all. In the end, as the narrative points out, even the combinatorics of the library itself is subjective, a phantasm of its inmates. On the one hand, the systems of order, structure and control inherent in the library are non-semantic. On the other hand they get continuously destroyed through semantics being read into them—theological semantics, for example since there are religious sects in the library holding certain beliefs about its sense and inner workings.

Borges' story renders combinatorics a purely speculative figure of thought. It is a speculation about a speculation: a speculation created around the speculative, subjective account of the first-person narrator. This self-destructive, paradoxical moment sets the story apart from a plainer, non-ironic speculation like Kuhlmann's. For Kuhlmann, there is still an identity of art, science, technology, philosophy, religion and cosmology. This leads to an integralist model of computation as totality. Computation does not yet imply, as in Borges, the negative of fragmentation and disintegration. Swift's fiction of the Grand Academy of Lagado stands precisely between Kuhlmann and Borges because it cuts into the totality of Lullist epistemology and, with its rationalist agenda, separates science from fancy, and observation from speculation. As a result, Lullist language computation is no longer part of serious science, but ends up in the realm of the obscure, occult, para-philosophical thought experiments, and from there in the realm of fiction and the arts. Only as arcane and speculative knowledge, is Lullism able to enter Borges' fantastic fiction.

Romanticist combinatorics

After Swift, combinatorial encyclopedism survives only in the niches of literature and speculative poetic science. In his 1798/99 encyclopedic fragment *Das Allgemeine Brouillon*,⁸ German romanticist poet and essayist Novalis (Friedrich von Hardenberg) sketches a new comprehensive integration of all science and knowledge on the grounds of an—itsself sketchy and fragmentary—“calculus.” It is a new attempt of a Lullist encyclopedia after Alsted, a speculative, essayistic encyclopedia which consists of fragmentary entries on knowledge in the light of a transcendental poetic philosophy. Its systematics takes inspiration from Leibniz' *mathesis universalis*, the project of a

⁸Novalis. *Das Allgemeine Brouillon*. Meiner, Hamburg, 1993 (1798/99). [72]

thorough mathematical language and description of thought. However, Novalis' "sketches" do not technically pursue the concept, but remain an idealist experiment. They shift combinatorics and computation from concrete, formal-technical manipulation of symbols—as before in 17th century proetic poetry—to a purely intellectual-reflexive, rather vague figure of thought: a meta-computational reflection of philosophical computability.

Computation as a romantic figure of reflection exists also in the *Livre*, the last project of French symbolist poet Stéphane Mallarmé.⁹ The book, which was never finished, should consist of ten volumes which could be shuffled at will. Like 17th century Lullism, and like the proto-Kabbalah of the *Sefer Yetzirah*, Mallarmé calculates the permutations of the *Livre*, as 3628800. The volume which did get published and is best known today is a visual-typographic poem, *Un coup de dès (A Throw of the Dice)*.¹⁰ It juxtaposes, so-to-speak, the mathematical permutation of the *Livre* with the random computation of a die that gets superimposed on its protagonist. A sailor, the poet's alter ego as it seems, navigates a ship in a storm. Having forgotten to make nautical calculations, he still refuses to throw the dice and give up to fate: "*Un coup de dès jamais n'abolira le hasard*," a throw of the dice will never abolish chance, is a key sentence of the poem. The last page imitates, by its typographic arrangement of words on the page, a starry sky showing the polar star with the little bear, culminating in the line "*UNE CONSTELLATION*," "A CONSTELLATION." It is, just as the mention of a book on combinatorial analysis in Borges' *Library of Babel*, a point where the text turns, recursively, into an index (or meta-data) of itself. After all, "constellation" references both the spatial arrangement of elements on the page and literally means the arrangement of stars in the sky.

With its double meaning, "constellation" rehashes the correspondence of macrocosm and microcosm known from Pythagorean, Neoplatonist and Kabbalist thought. It no longer does so in the context of a scientific worldview, but on the contrary, in self-chosen resistance to the modern scientific paradigm. Its formal experiment of integrating poetry, visual arts and musical composition into a Wagnerian total artwork is close to pre-modern art and thought such as Quirinus Kuhlmann's integration of combinatorics, cosmology, metaphysics and letter permutations on rotary dials.

⁹Jacques Scherer. *Le livre de Mallarmé*. Gallimard, Paris, 1977 (1957). [90]

¹⁰Stéphane Mallarmé. *Un coup de dès jamais n'abolira le hasard*. Gallimard, Paris, 1993 (1914). [66]

Concrete poetry

Mallarmé's poem inspired and instigated a whole experimental literary genre, the "constellations" of concrete poetry. Developed in the 1950s, they likewise stood for spatial arrangements of letters on pages, sometimes in conjunction with permutational poetic forms. Yet the philosophy and metaphysics of concrete poetry was quite contrary to Mallarmé. Its name was derived from Bauhaus artist and designer Max Bill and his coinage of "concrete art" in 1936. In the spirit of high modernism and functionalist architecture, most of concrete poetry sought to systematically reduce, rationalize and functionalize literature. A constellation of concrete poet Eugen Gomringer from 1969 reads:

no error in the system
 no reror in the system
 no rreor in the system
 no rroer in the system
 no rrore in the system
 no rror ein the system
 no rror ien the system
 no rror ine the system
 no rror in ethe system
 no rror in tehe system
 no rror in thee system
 no rror in the esystem
 no rror in the seystem
 no rror in the syestem
 no rror in the sysetem
 no rror in the systeem
 no rror in the systeem
 no rror in the systeem
 no rror in the systeem
 eno rror in the system
 neo rror in the system
 noe rror in the system
 no error in the system¹¹

Gomringer's poem is a strictly programmed permutational text: The error as embodied by the letter "e" shifts one position to the right in every next line. The algorithm stays intact throughout the whole

¹¹Translated from German from Eugen Gomringer. 3 variationen zu kein fehler im system. In Eugen Gomringer, editor, *konkrete poesie*, pages 63–64. Reclam, Stuttgart, 1972. [38]

poem despite its reference to an error. It could as well be implemented as a computer program. Indeed, Gomringer and his fellow concrete poets Claus Bremer and Tim Ullrichs created word permutational poems on computers in the early 1970s. “no error in the system” is tautological in two respects: First of all, the lines become redundant and repetitive as soon as one has grasped the algorithm. In contrast to a Proteus poem like Scaliger’s “perfade sperasti divos te fallere Proteu,” the algorithm is not being put down as a source code, but as its full execution. Secondly, there is no Gödelian moment like in Borges and Mallarmé, no self-reference which would make the system implode in a recursive paradox of text and context. There is, plainly, no error in the system Gomringer creates. The error—as the misplaced “e”—is visible on the first glance, but ceases to be an error in the light of algorithm that perpetuates it throughout the lines. There is an error in the system on the first glance, but no error in the system on the second glance so that the message of the poem proves right and therefore is superfluous. In Borges and Mallarmé there is on the contrary no obvious contradiction in the system on the first glance, but an abyss of epistemological paradoxes upon more thorough reflection.

Max Bense and “information aesthetics”

The poetics of concrete poetry and its constellations was first written down by Gomringer himself in his 1954 essay *From Line to Constellation*:¹²

Our languages are on the road to formal simplification, abbreviated, restricted forms of language are emerging. The content of a sentence is often conveyed in a single word. Longer statements are often represented by small groups of letters. Moreover, there is a tendency among languages for the many to be replaced by a few which are generally valid. Does this restricted and simplified use of language and writing mean the end of poetry? Certainly not. Restriction in the best sense—concentration and simplification—is the very essence of poetry. [. . .] The aim of the new poetry is to give poetry an organic function in society again, and in doing so to

¹²Eugen Gomringer. vom vers zur konstellation. In Eugen Gomringer, editor, *zur sache der konkreten*, volume 1, pages 7–12. Erker-Verlag, St. Gallen, 1988 (1954). [39], English translation from <http://www.ubu.com/papers/gomringer01.html>

restate the position of poet in society. Bearing in mind, then, the simplification both of language and its written form, it is only possible to speak of an organic function for poetry in terms of the given linguistic situation. So the new poem is simple and can be perceived visually as a whole as well as in its parts. It becomes an object to be both seen and used: an object containing thought but made concrete through play-activity (*denkgegenstand-denkspiel*), its concern is with brevity and conciseness. It is memorable and imprints itself upon the mind as a picture. Its objective element of play is useful to modern man, whom the poet helps through his special gift for this kind of play-activity. Being an expert both in language and the rules of the game, the poet invents new formulations. By its exemplary use of the rules of the game the new poem can have an effect on ordinary language.

The constellations were, in this sense, not strictly a computational text form; the definition of the poet as an inventor of "new formulations" which apply "rules of the game" however expresses a computational understanding of writing and literature. This idea found its theoretical underpinning in the philosophy of Max Bense, the major intellectual mentor of concrete poetry. Bense's theory, dubbed "information aesthetics," combined Claude Shannon's technical information theory with Charles S. Peirce's semiotics and philosophical aesthetics into a formalist, computational theory of modern art. By dialectical implication, it was also an aesthetic theory of computation based on formalist modern art. Formalisms in modern art, architecture and design, especially those in abstract painting and Bauhaus design, formed an important pretext for Bense's theory and its project of radically abandoning semantics.¹³ Classical semiotics still thought of signs having a meaning—for example through an artificial, abstract relation between sign and object ("symbol") through outer resemblance ("icon") or through the sign being a trace of the thing it represents ("index"). In 1948, Claude Shannon, a telecommunication engineer at the AT&T Bell Labs, coined a concept of information

¹³His extreme formalism was counterbalanced, one could argue, by his partisanship for artistic experimentation and play, and for avant-garde art in the 1950s, a period where particularly in Germany realism, existentialism and post-symbolist introversion dominated literature.

that did away with all semantics.¹⁴ It made information a technically quantifiable, measurable entity for determining (a) the transmission capacity of a channel and (b) the technical redundancy of data. These concepts, and Shannon's mathematical formulas, are still fundamental to digital information processing, for example for determining the throughput capacity of a network line and compression of data and files.

Bense wasn't interested in Shannon's theory as a pragmatic engineering perspective, but as a philosophical tool with which to reinvent aesthetics and poetics, stripping it bare, as he hoped, from any concept of meaning. His combination of semiotics with technical information theory was an artifice for turning the humanities and art criticism upside down. It resulted in a purist modernism which was ideological just in its ostensible refusal of ideology, and metaphysical in its radical refusal of metaphysics. After all, Bense's conflation of information theory with semiotics stripped computational forms of art and writing from their historical, intellectual implications. Mallarmé and other late romanticist poets like the German Stefan George had been the main sources of inspiration for the formal innovation of concrete poetry. Bense's theory however sought to look only at the form and cut all philosophical-metaphysical ties.

Pythagorean and occult metaphysics of macrocosmic and microcosmic equivalence had been pushed outside rational science since the 18th century. But not only romanticism remained heavily indebted to this tradition, but also, for example, the avant-garde word-play poetry of the Russian Futurist Velimir Khlebnikov. On the contrary, Bense and the post-war formalist avant-garde wanted to do away with metaphysics entirely and refound aesthetics on the grounds of modern science and technology. This move obscured the romanticist roots not only of concrete poetry,¹⁵ but also of Bense's philosophy itself. Like Novalis in his encyclopedic project, Bense took heavy inspiration from Leibniz' *mathesis universalis*, the post-Lullist attempt of unifying all human knowledge on mathematical grounds. Bense's move to clear the project from romanticist connotations and

¹⁴Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, July 1948. [92]

¹⁵In his preface to the anthology *konkrete poesie*, Reclam-Verlag 1972, Gomerlinger credits, among others, Mallarmé and sinologist Ernest Fenollosa whose essay *The Chinese Written Character as a Medium for Poetry* was a major influence on the poetics of Ezra Pound.

indeed any kind of overt metaphysics reflects, obviously, his experience with totalitarian politics of The Third Reich and Stalinist post-war East-Germany.

Bense's idea of formal programming as a model of an 'objective' art and philosophy ran in parallel to French linguistic structuralism of the 1950s and 1960s. It anticipated the so-called "linguistic turn" in the humanities of the late 1960s. The linguistic turn did not only happen in theory, but also in art in what critic Lucy Lippard described as the "dematerialization of art" around 1970. Pop art and Fluxus turned into concept art that, in the works of Joseph Kosuth, *Art and Language*, Lawrence Weiner and On Kawara for example, consisted only of written instructions. The first concept art show, curated by critic Jack Burnham in New York in 1970, accordingly had the title *Software*. It juxtaposed concept art works with experimental computer software development projects such as Ted Nelson's first prototype of a hypertext system. Burnham, a close collaborator of artist Hans Haacke before the latter turned from rigorous formalism to politically activist art, took his inspiration from cybernetics and general systems theory. The theoretical base was similar, but not identical to that of the continental European discourse of Bense and literary semiotics and structuralism.¹⁶ Burnham's ideas, too, imposed a rigorous scientific formalism onto art. They were more concerned with the visual arts than poetry though, and with the detachment of art from material objects more than with computational algorithmics. *Software* was, as Edward A. Shanken puts it in an essay on Burnham's exhibition, a "metaphorical premise," or device, that emphasized software as being different from hardware, not software as executable instruction code.

The underground art and activist journal *Radical Software* appeared under that metaphorical premise as well. Founded in the same year as the *Software* exhibition, it propagated an "Alternate Television Movement," juxtaposing aesthetic reflection with political debates about free media and publicly accessible radio spectrum, much like the contemporary free wireless network movement. In addition, it contained hands-on technical instruction for building and manipulating video equipment. Otherwise, the journal conceived of "software" purely as dematerialized art, and did not cover computer programming.

¹⁶Although a volume of Burnham's collected essays appeared, alongside the German translation of Abraham M. Moles' *Art and Computer*, in a German art book series under the title *Kunst und Strukturalismus (Art and Structuralism)*.

Modelled after the Bauhaus, Bense's older program turned poetry, concrete poetry in particular, into language design and art into visual design. These efforts were systematically pursued at the *Hochschule für Gestaltung* (*School for Design*) in Ulm, where Bense taught next to his regular professorship in nearby Stuttgart. *Hochschule für Gestaltung* was founded as a "second Bauhaus." Its director was Max Bill and faculty included former Bauhaus professors Johannes Itten and Josef Albers. Later in the 1990s, the *ZKM* media arts center nearby in Karlsruhe was established as yet another attempt of a new Bauhaus, this time as a "Bauhaus of Second Modernism" according to its initiators.

Situationism, Surrealism and psychogeography

In a counter-reaction to Bill, Bense and their functionalist thinking, Danish painter Asger Jorn, previously a member of a Surrealist splinter group in 1940s Paris, founded a *International Movement for an Imaginist Bauhaus* in 1956. In 1958, it became part of the Situationist International. In January 1959, the German section of the S.I. continued the opposition to Bense and Bill with a prankish attack on Bense in Munich: A public lecture of Bense was announced and once the audience had gathered, a tape recorder was switched on and the voice on the tape declared that Bense was unable to come and would instead give his talk in "cybernetic form." The talk was a deliberately nonsensical cut-up of German, Latin and French phrases with garbled quotations from Marx and Hegel. Yet the audience stayed through the lecture and applauded in the end. In the prank, the Situationists took Bense's cybernetic poetics and turned it as a tactical device against himself. The stunt displayed that his attempt to do away with semantics has its blind spot precisely in the semantics of his own statements that negated semantics. Secondly, it debunked the concept of technologically produced information as objective which the Situationists countered with a post-romantic and post-surrealist concept of aesthetic subjectivity. Like Jorn, the German Situationists attacked Bense, according to the S.I.'s report of its 3rd conference, for his "perfect continuation of constructivism." In his essay *Open Creation and its Enemies*, published in 1960 in the fifth issue of the journal of the

Situationist International,¹⁷ Jorn likened Bense to the concrete poetry-like French Lettrist poets, declaring him “the German equivalent of this anecdote of systematic, paradilectic, and deadly boring ‘Lettrist thought’.”

The flip-side of this critique was hostility of the Situationists to both artistic experimentation with new technology and philosophical reflection on computation. This hostility manifested itself particularly in the repeated Situationist attacks on communication theorist Abraham Moles (see p. 92). With their polemics against formalism and for “imagination,” the S.I. clearly continued the ideas of the French Surrealists who in turn were heavily indebted to romanticism. In his 1924 *Manifesto of Surrealism*, André Breton wrote that “We are still living under the reign of logic: this, of course, is what I have been driving at. But in this day and age logical methods are applicable only to solving problems of secondary interest.”¹⁸ His surrealism expresses laconic indifference to new technology: “Radios? Fine. Syphilis? If you like. Photography? I don’t see any reason why not. The cinema? Three cheers for darkened rooms. War? Gave us a good laugh. The telephone? Hello.”¹⁹ Along the lines of this technological scepticism, Surrealist “automatic writing” for example was not computational, but a psychic automatism that took the unconscious as its source code, not a calculus. It was still a foreign idea to Surrealism that computational formalisms could themselves be highly subjective and culturally coded, as the Pythagorean and Kabbalist tradition and the “semantics” of, for example, Lull’s “alphabetum” suggest. The Situationist concept of “psychogeography” had its roots in the aimless Surrealist drifts through Paris described in Breton’s 1928 novel *Nadja* and in Louis Aragon’s 1926 novel *Le Paysan de Paris*, and meant a purely subjective, para-scientific exploration of (chiefly) urban spaces through aimless drift. The surrealist drifts in turn were indebted to the romanticist “flâneur,” a wanderer “botanising the asphalt” as cultural theorist Walter Benjamin put it in his essay on 19th century poet Charles Baudelaire.²⁰

¹⁷Asger Jorn. La création ouverte et ses ennemis. In: Internationale Situationniste, editor. *Internationale situationniste. Édition augmentée*. Librairie Arthème Fayard, Paris, 1997 (1958-1969)., pages 175–196. [52], English translation online at <http://www.infopool.org.uk/6004.html>

¹⁸André Breton. Manifesto of Surrealism. In *Manifestoes of Surrealism*, pages 1–48. Ann Arbor Paperbacks, Ann Arbor, Michigan, 1924. [14], p. 9

¹⁹Breton [14], p. 46

²⁰Walter Benjamin. *Charles Baudelaire: A Lyric Poet in the Era of High Capitalism*. New Left Books, London, 1973. [8].

Computation and romanticist urban drift did not converge until the invention of “generative psychogeography” in the late 1990s through the Dutch artistic project <http://www.socialfiction.org>. Its *.walk* is a “psychogeographic computer,” operated by pedestrians who walk through street grids like electrons flow through the gates of computer chips. The *.walk* computer can execute simple program code like the following:

```
// Classic .walk
Repeat
{
1 st street left
2 nd street right
2 nd street left
}
```

Psychogeographic computing has a double effect: It demystifies computing, turning it into a radically simple and popular low-tech and low-cost operation. Secondly, in the spirit of Surrealism and Situationism, it liberates the imagination of what a computer can be and which purposes it may serve. Socialfiction.org thought has expanded and systematized this idea into a *University of Speculative Programming*, collectively editable Wiki website <http://twentiethcentury.com/uo/index.php/SpeculativeProgramming>. The site sketches the experimental potential of speculative programming as follows:

- pataphysical; pataphysics is a “science of imaginary solutions” according to its inventor, late 19th century pre-Surrealist novelist and dramatist Alfred Jarry. It could also be called a poetic, absurdist para- and anti-science. A pataphysical appropriation of computing factually exists in the work of the Oulipo poetry workshop (see p. 88);
- “casting spells on the OS,” reflecting magic as a forerunner of code execution;
- “social engineering,” especially in the form of artificial intelligence chat robots; which reminds of the S.I.’s tactical use of a manipulated tape recording as an analog Max Bense chat roboter. Since Alan Turing, chat robots are the fundamental to artificial intelligence research (see p. 106)
- questioning the traditional computer; thus, by implication also reflecting on the cultural history of computing aside from its materialization in electronic hardware.

“Speculative programming” reads as an attempt to sum up all philosophies and the complete cultural-imaginative history of computation,

including everything that is described in this booklet, too. Computing becomes a figure of thought and reflection not only in theory, but also in artistic practice. While the same could be said about Bense's philosophy, the implications are contrary. Instead of acknowledging subjectivity and imagination put into computations, computation becomes a token for a culture of scientific and engineering objectivity. Where Bense models art, criticism and aesthetics *after* computing, superimposing the latter on the former, speculative programming does the opposite. It models computation after the arts and and speculative imagination.

Markov chains

In Bense's "information aesthetics," art criticism turns, literally, into computation of data. Interpretation of meaning is substituted with formal analysis according to objectively quantifiable parameters, for example through word statistical analyses of writing. In Italo Calvino's 1979 novel "If on a Winter's Night a Traveller," the first person narrator, a writer, encounters a woman who refuses to read his novels, but instead feeds them as data into a statistical program:

She explained to me that a suitably programmed computer can read a novel in a few minutes and record the list of all the words contained in the text, in order of frequency. "That way I can have an already completed reading at hand," Lotaria says, "with an incalculable saving of time. What is the reading of a text, in fact, except the recording of certain thematic recurrences, certain insistences of forms and meanings? . . ." ²¹

Three subsequent pages of the novel are dedicated to the word statistics Lotaria computes. A footnote explains that Calvino took them from a computer-philological research work "Spogli elettronici dell'italiano contemporaneo" edited by linguist Mario Alinei in 1973.²² So Calvino's novel writes a parodistic critique of Bense's "information aesthetics" much like Swift early 18th century satire of Lullism. Claude Shannon's 1948 paper *A Mathematical Theory of Communication* describes such an algorithm for statistical text analysis:

²¹Italo Calvino. *If on a Winter's Night a Traveller*. Everyman Publishers, London, 1993 (1979). [24]

²²Mario Alinei and Alfredo Schiaffini. *Spogli elettronici dell'italiano delle origini e del Duecento*. Mouton, The Hague, 1968. [3]

“To construct [order-1 letter-level text] for example, one opens a book at random and selects a letter at random on the page. This letter is recorded. The book is then opened to another page and one reads until this letter is encountered. The succeeding letter is then recorded. Turning to another page this second letter is searched for and the succeeding letter recorded, etc. A similar process was used for [order-1 and order-2 letter-level text, and order-0 and order-1 word-level text]. It would be interesting if further approximations could be constructed, but the labor involved becomes enormous at the next stage.”²³

Shannon’s method is, in other words, to scan a text for the transition probability of letter occurrences, resulting in transition probability tables which can be computed even without any semantic or grammatical natural language understanding. (Otherwise, the program would require artificial intelligence.) This algorithm implements the stochastic model of Markov chains invented by the Russian mathematician Andrei Markov in 1906. Markov chains can be used not only for analyzing texts—and any other kind of information—, but also for recombining them based on the transition probabilities of their syntactical elements. In 1959, Theo Lutz, a computer scientist who collaborated with Max Bense at Technische Hochschule Stuttgart, processed phrases from Kafka’s novel *The Castle* with a Markov chain program. The result, called “stochastic Texts,” was published in Bense’s journal for contemporary experimental poetry. Shortly after, the first manifesto of the Oulipo (see p. 88) proposed to reinvigorate the old poetic collage form of the cento “by a few considerations taken from Markov’s chain theory.”²⁴

Independently from Bense, Lutz and Oulipo, critic and Joyce expert Hugh Kenner wrote, in collaboration with programmer Joseph O’Rourke, a text recombination program based on Markov chains. Dubbed *Travesty*, its source code was published in a 1984 issue of the popular computer magazine *BYTE*. For the algorithm, Kenner credited the “long-ago idea from the Father of Information Theory, Claude Shannon.” The code was adapted in 1990 by Larry Wall, creator of

²³Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, July 1948. [92]

²⁴François Le Lionnais. Lipo: First manifesto. In: Warren F. Motte, editor. *Oulipo. A Primer of Potential Literature*. University of Nebraska Press, Lincoln, London, 1986., pages 26–28. [61], p. 27

the Perl programming language, and published as a programming example in the first edition of the book *Programming Perl*.—The second edition of the same book featured examples of “Perl poetry”²⁵ (see p. 94)—With poet Charles O. Hartman, Kenner co-authored *Sentences*, a volume of poems generated with the help of *travesty*.²⁶ In 1994, experimental poet and former Fluxus member Jackson MacLow generated a number of his *42 Merzgedichte in memoriam Kurt Schwitters* with the Kenner’s *travesty* program and, one year later, Austrian contemporary composer Karlheinz Essl reassembled a Bach violin sonata through Markov chain computation, calling it “Bach sausage.” Likewise, the DOS-based poetic language manipulation toolkit *POE* designed by Austrian experimental poets Ferdinand Schmatz and Franz Josef Czernin (see p. 112) includes a Markov chain function. Around the same time, numerous popular Markov chain-based text manipulation programs were written, such as Dissociated Press, a standard function of the GNU Emacs text editor, TextMangler and Deconstructor for MacOS, dadadodo by Jamie Zawinsky, the former project leader of the Mozilla web browser, or Mark V. Chaney for DOS.

Since Shannon’s *Mathematical Theory of Communication*, Markov chains have arguably become the third most popular algorithm for computational text generation, next to permutation and recursion. Markov chains are a strictly analytical, not synthetic method, being agnostic to the data they process. They work on arbitrary input, contrary to older synthetically combinatorial methods such as Lull’s *Ars* and Renaissance proteus poems which always processed fixed, pre-inscribed words. While Lull’s *ars* abstracts data from algorithms through its separation of the *tabula* from the *figurae*, the *figurae* are designed to process only the particular, immutable elements of the static *tabula*. Even Swift’s and Borges’ dystopias of language computing, with their ontological randomness and contingency, still rely on a fixed data set, the alphabet.

Tristan Tzara and cut-ups

One could believe that poetic computations of mobile data sets could not be imagined before modern computers were invented. However,

²⁵Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl*. O’Reilly, Cambridge, Köln, Paris, Sebastopol, Tokyo, 1996. [99]

²⁶Charles O. Hartman and Hugh Kenner. *Sentences*. Sun and Moon Press, Los Angeles, 1995. [45]

the first modern art work based on a computational process and arbitrary input data dates back to 1923. Tristan Tzara's advised *To Make a Dadaist Poem* as follows:

Take a newspaper.
 Take some scissors.
 Choose from this paper an article the length you want to make your poem.
 Cut out the article.
 Next carefully cut out each of the words that make up this article and put them all in a bag.
 Shake gently.
 Next take out each cutting one after the other.
 Copy conscientiously in the order in which they left the bag.
 The poem will resemble you.
 And there you are—an infinitely original author of charming sensibility, even though unappreciated by the vulgar herd.²⁷

What Tzara describes here is an algorithmic process, in fact, a simple computer creating random permutations of arbitrary input. For the first time after the romantic period, poetics defies the concept of the genius and turns into formal instruction again; formal to the point where it can be mechanized. The 18th and 19th century ridicule of formalisms in composition and art that began with Swift is taken up here and turned against itself. Tzara's "infinitely original author of charming sensibility, even though unappreciated by the vulgar herd" clearly mocks the romantic genius.

While the Dadaist poetic algorithm is the same as the word permutations in the *Sefer Yetzirah* and in proteic poetry since Optatianus and Scaliger, its imagination and ideology is reversed: There is chaos instead of order, fragmentation instead of totality. It also lacks, on the first glance at least, any metaphysics and cosmology of macro- and microcosm. If one juxtaposes the Dadaist poem to Mallarmé's *Livre*, then it reads as the latter's radical other; it uses a similar algorithm, by implication also a similarly free form typography, but contradicts both the romanticist aesthetics of the genius and the representation of a Pythagorean macrocosmic order in the microcosm of the artwork.

²⁷Tristan Tzara. Pour faire une poème dadaïste. In *Oeuvres complètes*. Gallimard, Paris, 1975. [98], English translation online at, among others, <http://www.poets.org/viewmedia.php/prmMID/5774>

Still, the ostentative nihilism of Tzara's random computation creates another metaphysics. Gysin's and Burroughs' "Cut-ups" for example are a straight-forward adaption of Tzara's method. Burroughs' essay *The Cut-Up Method of Brion Gysin* acknowledges this in its first sentence:

At a surrealist rally in the 1920s Tristan Tzara the man from nowhere proposed to create a poem on the spot by pulling words out of a hat. A riot ensued wrecked the theater. [...] In the summer of 1959 Brion Gysin painter and writer cut newspaper articles into sections and rearranged the sections at random.²⁸

However, the cut-up differs from Tzara's Dadaist poetics in several respects. In the same text, Burroughs refers to the "place of mesaline hallucination: seeing colors tasting sounds smelling forms." As a magical and ecstatic technique, the cut-up resuscitates the old metaphysics that Tzara seemed to have done away with.

John Cage's indeterminism

The same can be observed in the adaption of the Dadaist random montage music in the "indeterminist" music of John Cage his contemporaries and students from Earle Brown, Morton Feldman, Christian Wolff to the Fluxus movement. In John Cage's piece *Atlas Eclipticalis*, an astronomical map serves as a superficially random score, and establishes, once again, a macrocosmic-microcosmic correspondence which justifies a randomness of art through a higher-order randomness of nature. Cage also uses the Chinese *I Ching*, the *Book of Changes*, as a compositional algorithm for a number of his works, among them *Music of Changes* and *Roaratorio*.²⁹ For these purposes, he later used a computer software adaption of the called *IC*, written in C for DOS by Andrew Culver.³⁰ Cage replaces the cosmological-mathematical order of Pythagorean Western music with a metaphysical anarchism. This anarchism nominally draws from Eastern philosophy, but in fact relied only on superficial studies—through the attendance of a few evening lectures—of the non-traditional, highly Americanized Zen of Daisetz T. Suzuki.

²⁸William S. Burroughs. *The Cut-Up Method of Brion Gysin*. In: William S. Burroughs. *The Third Mind*. Viking, New York, 1978. [17]

²⁹John Cage. *Roaratorio. Ein irischer Circus über Finnegans Wake*. Athenäum, Königstein/Taunus, 1982. [20]

³⁰The program is still available under <http://www.newmus.net/apps/iching.exe>.

Just like Tzara's Dadaist poem manifests an anti-romanticism that nevertheless shares many formal traits with late-romanticist literary experimentation, Cage's approach to musical composition could be seen as anti-romantic and ultimately anti-Western while owing more to Western tradition than it acknowledged. Cage's anti-tradition is most pronounced in *Credo in Us* from 1942, a piece that pokes fun at late romanticist symphony music by playing it randomly from gramophone records. It could be called the first musical piece using "scratching" and "Plunderphonics." Still, Cagean indeterminism remained reciprocal to the mathematical determinism of Western music. It only replaced the Pythagorean formula of symmetry through a formula of chance. It seems only logical that Cage formalized his composition methods and implemented them into computer software tools. He used 24 custom-written computer programs for his composition.³¹ Cage wrote computational poetry, too. With his method of "reading through,"³² he compressed literary works like Joyce's *Finnegans Wake* to a fraction of their original size. The algorithm he used were mesostichs, the retrieval of words in a text whose middle letters match the letters of a name; electronic poet Jim Rosenberg wrote him a custom computer program *Mesolist* for this purpose.³³

It seems paradoxical that Cage sought to create indeterminacy through algorithms. While this contradiction superficially resembles the link of algorithms and the imaginary in psychogeographical computing, it in fact is more naive since it relies on a dubious understanding of "chance." Algorithms can be used for chance operations as Tzara's Dadaist poem shows, and—with much older origins probably in ancient India—the dice as a random computing device. But such random operations create stochastic chance, not philosophical-ontological chance. Throwing a die is a stochastic chance operation with the possible outcome of 1, 2, 3, 4, 5, 6. Since these results are foreseeable as the set of potential results, they represent not an ontological, but a deterministic chance. Mallarmé describes it precisely in his sentence: "throwing the dice never abolishes chance." Ontological chance, and therefore true indeterminacy, would occur if the die would crack, vanish, or show the number seven. With pieces like *Atlas Eclipticalis*, *Variations I/II* and all *I Ching*-based compositions, it seems as if Cage misreads stochastic chance for ontological

³¹According to Culver's website <http://www.anarchicharmony.org/People/Culver/CagePrograms.html>.

³²John Cage. *Roaratorio. Ein irischer Circus über Finnegans Wake*. Athenäum, Königstein/Taunus, 1982. [20] contains a chapter *Reading through Finnegans Wake*

³³Referenced on Andrew Culver's website

chance. The mere fact that a musical piece sounds “like John Cage” or “Cagean” because of its atonality and lack of development for example, disproves Cage’s claim of “indeterminacy.” If his music were really, ontologically indeterminate, it should—for example—be able to sound like a Britney Spears song, too.

This example illustrates a general limit of computation and software. There is, first of all, no way of turning them into anti-formal, anti-determinist technology, either through stochastic chance, disruption of semantics (as in the cut-ups) or a higher-order complexity of programming (as in artificial intelligence, see p. 106). Secondly, the formalisms are always meaningful. In the case of Cage, they seem to subvert the intended anarchist ontology of chance—as it is grounded in 20th century aesthetics, philosophy and religion—into a stochastic determinism. This unveils the blind spots of Cage’s compositional approach and the reason why an “indeterministic” chance composition of Cage can sound strikingly similar to totally deterministic serial compositions of Stockhausen or Boulez. If one compares Cage and Stockhausen and their seemingly contrarian understanding of musical composition, one can practically observe how deterministic over-complexity turns into chaos and conversely, chaos ends up as a formalism and cliché.

Tzara’s poem struggles with this problem only to a lesser degree. As a parodistic and cynical device, it does not have to match up to an ontological indeterminism. The relative lack of cynicism is what distinguishes Cagean aesthetics from Dadaist provocation. Neither do Burroughs’ cut-ups run into the trap of deterministic chaos because they are tools for reaching another, magical, hallucinatory and ecstatic order, and use language to overcome it.

In addition to permutational shuffling, the cut-ups frequently employ recursion, the processing of something through itself, as their poetic method. In *Cut-Ups Self-Explained*, the last paragraph of Burroughs’ *The Cut-Up Method of Brion Gysin* applies the method—or algorithm—to its own instruction:

ALL WRITING IS IN FACT CUT-UPS OF GAMES AND
ECONOMIC BE HAVIOR OVERHEARD? WHAT ELSE?
ASSUME THAT THE WORST HAS HAPPENED EX-
PLICIT AND SUBJECT TO sTRATEGY IS AT SOME

POINT CLASSICAL PROSE. CUTTING AND REARRANGING FACTOR YOUR OPPONENT WILL GAIN INTRODUCES A NEW DIMENSION YOUR STRATEGY. [...] ³⁴

But what is the practical effect of this self-processing of text? As in Cage's conflation of stochastic chance and ontological indeterminacy, there seems to be a surplus of imagination projected into the formalism; as if texts were becoming lucid, pronouncing hidden truths and achieving occult effects through their underlying formal operations. However, the results perpetually fail to match up to those expectations. The recursive processing of the cut-up instruction through itself at least points out a paradox: within a permutative, random or stochastic process that mobilizes and rearranges words, the algorithm itself remains an immutable center. A random poem like Tzara's is *not* random to the extent it relies on a clearly defined, fixed algorithm. It can digest and transform all writing with the singular exception of itself. Otherwise, it would destroy its own instruction and with it the digestion and transformation of writing. Once again, the problem remains that in any running computational program, the instruction—or game rule—remains, internally, a formalism that is not part of the game.

Italo Calvino and machine-generated literature

In his 1967 lecture *Cybernetics and Ghosts*,³⁵ Italian novelist Italo Calvino concludes that computer-generated poetry tends to be “classicist;” classicist in the understanding of Italian, French and Spanish literary history, as a poetry governed by strict normative poetics and rules of form. In spite of this restriction, he sketches a general model of language and narration as computations. Calvino speculates that early humans had a limited repertory of sounds and words and therefore needed combinatorics to expand a scarcity of vocabulary into a richness of communicative means. It follows for him that literature is a “combination game.” He, the poet, can be replaced by a “mechanical device.” In his reasoning, Calvino cites the linguistic and semiotic structuralism of ethnologist Claude Lévi-Strauss and Russian philologist Vladimir Propp. This reference reflects Calvino's exposure to the structuralism of the Tel Quel group around Julia Kristeva and Roland Barthes whose meetings he attended in Paris.

³⁴William S. Burroughs. The Cut-Up Method of Brion Gysin. In: William S. Burroughs. *The Third Mind*. Viking, New York, 1978. [17]

³⁵Italo Calvino. *Cybernetics and Ghosts*. In *The Uses of Literature*, pages 3–27. Harcourt, San Diego, 1982 (1967). [22]

In his 1927 book *Morphology of the Folktale*, Vladimir Propp concluded from a structural analysis of Russian fairy tales that their plots could be reduced to one universal formula.³⁶ This formula denotes both possible combinations and variations. It is, in fact, an algorithm which could be adapted as a computer program for generating arbitrary fairy tale plots. Propp's formalist philology appears like a straightforward continuation of Lullist encyclopedism, the exhaustion of a field of knowledge through combinatorial means. Vice versa a Lullist language generator like Harsdörffer's *Denckring* manifests proto-structuralist linguistics. In both Harsdörffer and Propp, synthetic totalism and analytical fragmentation are two sides of the same coin, and their algorithms can be used for either.

As a device of analysis, Propp's morphology first of all gives insight into the formulaic, constructed nature of folk tales. It thoroughly debunked the 19th century romanticist view of the folk tale as an irregular and chaotic manifestation of popular fantastic imagination (which was superior to highbrow art). The same romanticist idea continued in surrealism, with its fondness of popular culture and trash—like, for example, the *Grand Guignol* gore theater shows—and to some degree in Situationism with its detournement of comic strips and martial arts movies (like René Vienet's *Can Dialectics Break Bricks?*). With its discovery of a popular culture which is so formulaic that it is computable, Propp's analysis could on the other hand be considered a forerunner of psychogeographical computing with its populist low-tech approach to programming. With their emphasis on popular imagination and popular cultural practices, both Propp and psychogeographical computing themselves owe to romanticist thinking. Italo Calvino's *Invisible Cities*, a novel which describes imaginary cities and generates them through a combinatorics of fantastic attributes, manifests another intersection of computation and romantic urbanism.³⁷

Software as industrialization of art

Movie plot generators. In a newspaper essay *Make Your Own Movie* which appeared in the book *Misreadings*, Umberto Eco turns Propp's combinatory plot generation and, by implication, structuralist morphologies into a parody. He proposes a number of algorithms for generating movie plots, each of which sums up the stereotypical mannerisms of the prototypical angry young film director and

³⁶Vladimir Propp, editor. *Morphology of the Folktale*. University of Texas Press, Austin, Texas, 1968 (1927). [78]

³⁷Italo Calvino. *Invisible Cities*. Harvest Books, Fort Washington, 1978. [21]

the filmmakers Michelangelo Antonioni, Jean-Luc Godard, Ermanno Olmi and Luchino Visconti.³⁸ The Antonioni algorithm works as follows:

An^x empty^y lot.^z She^k walks away.”

Variants Key

x: Two, three an infinity of. An enclosure of. A maze of.

y: Empty. As far as the eye can see. With visibility limited due to the sun’s glare. Foggy. Blocked by wire-mesh fence. Radioactive. Distorted by wide-angle lens.

z: An island. City. Superhighway cloverleaf. McDonald’s.

[...]

Eco’s theoretical works assume a rationalist and scepticist position within European semiotics and structuralism, rejecting ontological concepts of structure like those of Lévi-Strauss. In the same spirit, this parody pokes fun at an ontologization of structure. It also dismantles the predictable elements and clichés of 1960s/70s European “auteur cinema” with its perpetuation of the romanticist author-genius. When Eco wrote the text in 1972, he couldn’t foresee that a very similar recipe was to be used, twenty years later, in the commercial software *Plots Unlimited*, a PC program for screenwriters which generates plot lines out of an internal database of plot elements.³⁹ Since *Plots Unlimited* was published as a book with cross-referenced numbered paragraphs, its algorithm is transparent; it’s the same combinatorymorphological formula of Lull, Harsdörffer, Propp and Eco, using only a more elaborate set of data elements. With its title alone, *Plots Unlimited* puts itself into the tradition of encyclopedic combinatoric poetics, but does so as a commercial and pragmatic industrial tool. It is naive, and not sarcastic like Eco’s combinatorics. That, though, does not change its implication that much of contemporary movies and television soap operas might be based on its formulas, product of the *Plots Unlimited* matrix, so-to-speak. (The program is in fact a popular tool among screenwriters.) Much of popular culture could be machine products in a more literal sense than even Frankfurt School

³⁸Umberto Eco. *Misreadings*. Harvest, San Diego, 1993. [32], p. 145-155

³⁹Tom Sawyer and Arthur David Weingarten. *Plots Unlimited*. Ashleywilde, Malibu, 1994. [88]

sociologists Adorno and Horkheimer imagined when they coined the term “culture industry” to describe Hollywood in the 1940s.⁴⁰

Authorship and subjectivity

Cornelia Sollfrank’s *Net.art Generators*. Perhaps the most consequent, if ironic, project of industrializing art was Andy Warhol’s “factory” art. It industrialized its production, aesthetics and even the identity of the artist: Warhol’s art was collectively and industrially produced, used industrial mass media images as its material, and factory members with silver wigs appeared as Andy Warhol for public lectures. Through its seriality, his work resembles machine-generated art and later proved to be ideal source material of artistic computations. In 1997, artist Cornelia Sollfrank developed, in collaboration with a number of programmers—Ryan Johnston, Luka Frelih, Barbara Thoens, Ralf Prehn, Richard Leopold—, a series of *Net.art Generators*, web-based software programs that retrieve websites according to user-entered search terms, reassembling them as digital collages. Sollfrank was not interested in an autonomous generative art, but in the political and philosophical issues the net.art generators create. She and her programmers, for example, did not anticipate that the generators would create endless variations of Andy Warhol’s flower pictures (figure 3)—which in turn are based on a botanical photograph by American photographer Patricia Caulfield from 1962. Next to the many unauthorized variations of the Warhol flowers circulating as postcards and poster prints, the computer-modified flowers created unforeseen questions of authorship and originality. No dubious “artificial intelligence” (see p. 106) was necessary to create these issues. It was sufficient that the artwork was transformed from a solid entity to an automatic process. Moreover, it was designed by multiple people and working on arbitrary input data. As a result, an exhibition of Sollfrank’s net.art generators and their Warhol variations was cancelled by the organizers out of fear of being sued for copyright violation.

What Tzara’s Dadaist poem puts into ironic terms, the “charming sensibility” or subjectivity of the artist who creates a work according to an algorithm, is at the center of Sollfrank’s philosophical and legal reflections. Who exactly is the creator of a Warhol flower variation computed by the net.art generators? Caulfield as their original photographer, Warhol as their first artistic adopter, Sollfrank as the artist

⁴⁰Theodor W. Adorno and Max Horkheimer. *Dialectic of Enlightenment*. Verso, London, 1979 (1947). [1]



FIGURE 3. Variations of Andy Warhol's *Flowers* created by a *Net.art* generator

who created the concept of the net.art generators, the programmers who technically designed and implemented them, the users of the net.art generator, or the running program itself?

Conventional software for artists still sells the idea of the artist as an autonomous creator who works with the aid of, but isn't replaced by, algorithms. Artistic generators like those of Sollfrank reverse the model. They redefine authorship as the artistic design of an algorithmic process and, once this process is set into motion, the observation and reflection of its effects. The industrial nature of its results puts into question traditional categories of authorship, originality and genius. This approach builds and expands on the questions and provocations Tzara's Dada poem instigated. Programmer Richard Leopold who wrote one of Sollfrank's generators credits Dada as its main inspiration and calls the generated pages "Dada content" (but employs Markov chains instead of permutation as the generative algorithm).

What sets apart Sollfrank's generator from Tzara's poem however is that it explores its philosophical, aesthetic, legal and political implications in a more rigorous and systematic way. The idea that artists turn into designers and philosophical explorers of computations, paying only secondary attention to the output of the process per se, contradicts and subverts the industrial logic of the artistic software tool. Even *Plots Unlimited*, a rare example of computational-generative poetics within a commercial software package, still clings to the myth of being a transparent "aid" to the artist. Its advertising literally says that "with *Plots Unlimited* you'll develop your *own original* material

[...]—stories that sell [emphasis as in the original text].”⁴¹ Other computer programs for artists disguise themselves as mere tools in the hands of artists in more subtle ways. Word processors, graphics editors, desktop publishing and musical composition software are all based on two cultural premises:

- (1) To emulate, in look and feel, analog tools—typewriters, paint brushes, layout desks, scores and pianola rolls;
- (2) To pass themselves off as “transparent” tools, i.e. technology that obeys the user.

The latter is a fiction to the degree that those tools create their own aesthetics and have their own implied politics.

Software involves interface paradigms with encoded cultural pre-conceptions of what, for example, a “document,” “writing,” “designing” is. It has embedded concepts of the order of things, of communication and workflows. To this extent, software controls its users. Yet it sells the illusion that the user is fully in charge. Since the early 1990s, pop cultural graphic design has largely been driven by short-lived fashionable gimmicks and plug-ins in programs like . The whole musical genre of bootleg pop remixes would not exist without the programs *Acid* by Sonic Foundry and *Traktor* by Native Instruments. Even these programs maintain the fiction of transparency and user control, excluding algorithms that do not just aid, but actually generate work. Among the few exceptions are *Plots Unlimited* and *Band-in-a-Box*, a program that automatically creates musical arrangements.

Signwave Auto-Illustrator. The taboo of “transparent” software—to not openly interfere with the artist—is systematically addressed and subverted in Adrian Ward’s computer program *Signwave Auto-Illustrator*.⁴² It transforms vector graphics software into a generative program with an agenda of its own, or rather, of its programmer who codes his subjectivity into algorithms. With a user interface that precisely mimicks the commercial graphics program Adobe Illustrator, Auto-Illustrator implements, for example, a text tool that writes its own, randomly generated texts. Other functions turn artwork into “instant Bauhaus,” leave “bugs” that wander around in the illustration, or render circles as smilies. However, the program is functional. It generates proper graphics files and has been practically employed for the graphic design of flyers and record

⁴¹Tom Sawyer and Arthur David Weingarten. *Plots Unlimited*. Ashleywilde, Malibu, 1994. [88], cover text

⁴²<http://www.signwave.co.uk>

covers. *Auto-Illustrator* combines encyclopedism and fragmentation as anticipated by Lullism and Dadaism respectively. They do not contradict, but complement each other in the program. In the many years of its development since 2000, *Auto-Illustrator* has acquired an encyclopedic wealth of features, similar to commercial software that, incorporating more and more functions, strives to be the ultimate tool for its purpose. *Auto-Illustrator* openly renders the same totality absurd by accumulating eccentricities and personal fancies.

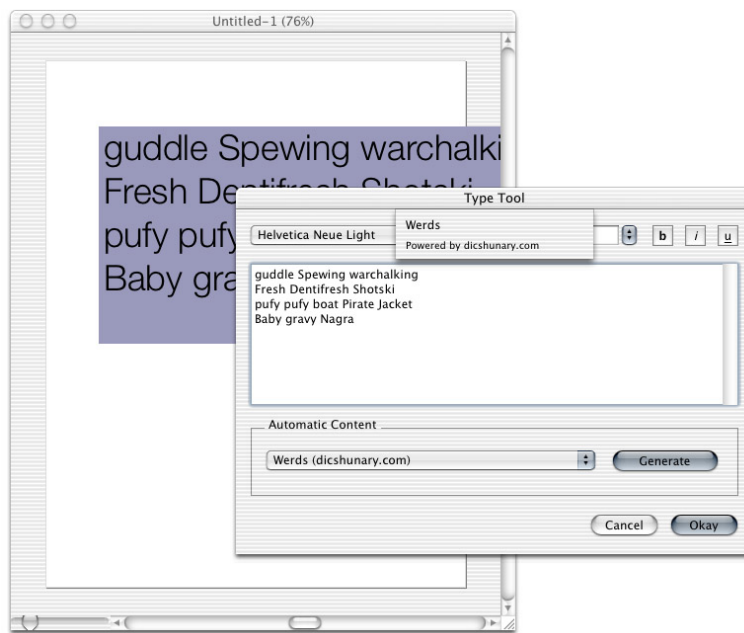


FIGURE 4. Random text tool of *Auto-Illustrator*

Algorithms as subjective expression were not conceived of in classical computational poetics. Both occult and scientific computations, be it magic, Kabbalah, encyclopedism or algorithmic calculus, rely on the idea that computation expresses a higher objective order; be it divine law, or the laws of logic and mathematics. It's hardly surprising that computation often oscillated between the occult and scientific poles, blurring its boundaries: from Pythagorean mathematics to eventually metaphysical computations such as those of Karlheinz Stockhausen and John Cage. Nevertheless, as the "semantics" of formal languages, i.e. the choice of their cultural denominators like Lull's *alphabetum* indicates, computations were never able to

do without superimposed meaning, inscribed subjectivity, embedded metaphors.

Since the 1970s, the field of “generative art” relies on the same illusion of objectivity through computation. The most comprehensive theoretical foundation for this field is provided by Max Bense’s writings. Even today, media theory remains centered around and preoccupied with the idea of an anti-subjective, post-human autonomy of machine processes. The roots of this media theory prominently lie in McLuhan’s statement that the medium is the message—i.e. the technology itself bears the meaning instead of being its neutral purveyor—and Heidegger’s philosophy of matter and technology. These theories neglect that the machine processes were designed by humans in the first place and can be seen on the contrary as embedding subjectivity into formulas, processes and hardware. When Cornelia Sollfrank states that “a clever artist makes the machine do the work,” it still implies that the artist *makes* it work in the first place.⁴³ Poiesis, making, becomes a second-order poesis of making something that makes something else. So poetry, making, turns into poetics, the making of making. When making turns into meta-making, subjectivity simply shifts to a second order position, residing in the formula instead of the product. This fact is being repeatedly ignored by critical observers whose perspective remains fixated on the product and who wrongly conclude, in a fallacy reminiscent of Plato’s cave, that technology has done away with the subject behind the work.

Calvino’s lecture on *Cybernetics and Ghosts* ends its theory about language and poetry as a “combination game” with the speculation that one day, the machine might be able to defy its own rules. This utopian hope has been, throughout the history of arts and computing, frustrated again and again (see chapter 4). It is an ultimately magical hope: That computing may one day transcend formalisms, and thereby its own technical grounds and limitations, in a moment of ontological chance similar to that of a die displaying the number 7. The same utopian hope drives artificial intelligence research, artificial life research and “virtual reality” computing. It also underlies the expectation of more “humane,” “fuzzy” computer interfaces which began with the invention of the graphical user interface at Xerox PARC and its popularization with the Apple Macintosh.—The “Humane Interface” was the last, unfinished software development project of Macintosh co-creator Jef Raskin before his death in 2005.—The same hopes

⁴³Cornelia Sollfrank, editor. *net.art generator*. Verlag für moderne Kunst, Nürnberg, 2004. [94]

and expectations drove the mainstream of digital “media art” and media theory in the 1990s. More critical artistic works, notably those of Jodi and I/O/D (see p. 117 and 95) next to Sollfrank’s and Ward’s, have questioned these utopias and contrasted them with the critical insight that

- computation remains, technically, a formalism;
- contrary to the assumptions of “artificial intelligence,” “virtual reality” and company, semantics can be had, and already exists, even without transcending formalisms or making them complex.
- formalisms, on the contrary, have a cultural semantics of their own, even on the most primitive and basic level. With a cultural semantics, there inevitably is an aesthetics, subjectivity and politics in computing.

Pataphysics and Oulipo

Alfred Jarry and the Collège de Pataphysique. The insight that even the most simple formalism has a cultural impact resolves the old conflict between computational poetics as higher-order, metaphysical or scientific objectivity on the one side and subjectivist aesthetics on the other. Since Jonathan Swift, empirical and aesthetic thinkers opposed computation precisely on those premises. Romantic technology, such as Surrealist games, Situationist and computational psychogeography, is another model of resolving the opposites. The contradiction between “objective” science and eccentric subjectivism however is most comprehensively subverted, through a simple clashing or “discordia concors” of the two poles, in “mad” or “poetic science.” Pataphysics was coined by its inventor, 19th century absurdist French dramatist and novelist Alfred Jarry as a “science of imaginary solutions.” His 1898 novel *Dr. Faustroll* defines pataphysics as the “science that added to metaphysics, either in itself or outside itself, and extend as far beyond metaphysics as the latter extends beyond physics.”⁴⁴ With this humorous foundation, pataphysics is, factually, a poetic science. Unlike previous poetic sciences and epistemologies such those of Novalis and Mallarmé, it does not take refuge in older metaphysical and theosophical paradigms of macrocosmic and microcosmic analogy. It is a poetic science that may, as Jarry’s novel shows, be founded on empirical absurdism, but in general exists outside the categories of rationalist physics and theosophical metaphysics.

⁴⁴Alfred Jarry. *Exploits and Opinions of Dr. Faustroll, Pataphysician*. Exact Change, Berkeley, 1996 (1911). [51]

Founded in 1949 by among others Raymond Queneau, Michel Leiris and Boris Vian, the *Collège de Pataphysique* continued a Parisian-French avant-garde tradition at whose center previously had been André Breton's surrealist group. Queneau had been a member of surrealism for a short period, and satirized it in his novel *Odile*. His chief fields of interests were mathematics and linguistics, and his day job was to be the editor of encyclopedias in the Editions Pléiade. Perhaps under his influence, the Collège de Pataphysique abandoned the romanticist, communist-political and psychoanalytical pretext of surrealism. This legacy was continued instead by the Situationist International which consequently criticized pataphysics as a "new religion in the making."⁴⁵

Raymond Queneau's 100,000 Billion Poems. In 1960, the *Collège de Pataphysique* founded its own literary chapter, the Oulipo (*Ouvroir de littérature potentielle*, "Workshop of Potential Literature"). Its core members were Raymond Queneau and mathematician François Le Lionnais. Earlier in 1947, Queneau had spelled out his own obsessions with mathematics, encyclopedism and street slang in the *Exercices de Style*, a narrative of one short everyday scene in 99 different stylistic variations.⁴⁶ In 1961, he extended this concept into a computational poem, the *100,000 Billion Poems*, a combinatory sonnet in ten variations.⁴⁷ It was printed in a book whose lines were individually sliced so that each line of poem could be turned like a page and picked from ten alternatives. From ten alternatives for the twelve sonnet lines, 10^{12} possible poem combinations result. Queneau subverts the rigorous classicism of the sonnet form and its Alexandrine meter through slang colloquialisms and through stereotypes sarcastically perpetuated in the poems. His preface credits the playful form of the book to children's books and disclaims any influence from surrealist games. The encyclopedism of his 100,000 billion poems is one of a perpetuated proverbial, trivial wisdom. As a pataphysical work, they contradict modern science through their combination of a Lullist synthetic scholasticism with an empirism of proverbial clichés. Queneau remarks that

⁴⁵This is the title of a text Asger Jorn published in the 6th issue of the Situationist International journal in 1961, *Internationale Situationniste*, editor. *Internationale situationniste. Édition augmentée*. Librairie Arthème Fayard, Paris, 1997 (1958-1969). [93].

⁴⁶Raymond Queneau. *Exercices de style*. Gallimard, Paris, 1947. [81]

⁴⁷Raymond Queneau. *Cent mille milliards de poèmes*. Gallimard, Paris, 1961. [82]

Dans un ordre plus abstrait, [. . .] l’Absolu, selon Jarry, “les clichés sont l’armature.” (In an abstract order, [. . .] clichés are, according to Jarry, “the armature of the absolute.”)

On its first page, the book quotes Alan Turing, and an afterword written by mathematician and pataphysician Le Lionnais cites one of Georg Philipp Harsdörffer’s proteic poems along with the music of John Cage and Stockhausen. Written in an ironic tone, it contains a straightforward literary history of poetic and artistic computations between the lines. Furthermore, this afterword bears the title “À propos de la littérature expérimentale” (*Concerning Experimental Literature*).⁴⁸ A forerunner of the Oulipo, created by Queneau and Le Lionnais in 1959, was called “Séminaire de Littérature Expérimentale” (*Seminar of Experimental Literature*). With the renaming of the project into Oulipo, the concept of “experimental literature”—bearing Le Lionnais’ formalist signature—was dropped, too.

Algorithmics as constraint and anti-formalism. Oulipo’s combination of poetry and mathematics was neither strictly rationalist, nor formalist. In an essay *Potential Literature* that appeared in 1964, Queneau clarifies that

We are not are not concerned with experimental or aleatory literature (as it is practiced, for example, by Max Bense’s group in Stuttgart).⁴⁹

Instead, Queneau explains, Oulipo is concerned with playing unpretentious games. He characterizes its poetics as “naive,” “craftsman-like” and “amusing.” Along these lines, the poetic formalisms developed by Oulipo do not employ algorithms as an expansion of language, but literally, and with darker humor, as “constraints.” This sets Oulipo apart from the poetic Lullism of the 17th century, the cut-up poetics of Burroughs/Gysin and the “artificial poetry” of the Stuttgart School, all of which understood poetic computations as a means of transcending the limitations of human creation and subjectivity.

Oulipo conceived of formalist poetics as a game-like artificial restriction on writing. As in a game where players have to find clever

⁴⁸Raymond Queneau. *Cent mille milliards de poèmes*. Gallimard, Paris, 1961. [82]

⁴⁹Raymond Queneau. *Potential literature*. In: Warren F. Motte, editor. *Oulipo. A Primer of Potential Literature*. University of Nebraska Press, Lincoln, London, 1986., pages 51–64. [83], p. 51

solutions around the hurdles superimposed by the rules, the algorithmic restrictions had to be compensated by imagination. A whole set of formalisms was created, such as lipogrammatic constraints which prohibit the use of certain letters. Poetic imagination was challenged to find a creative circumvention around the constraint. Oulipo member Georges Perec, for example, wrote a bulky novel *La Disparition* (English title: *A Void*) without a single occurrence of the letter “e.”⁵⁰ Oulipo’s alleged naiveté was its sophistication. It neither bought into the fallacy of scientific objectivity and machinic intelligence like Bense’s group, nor did it regress into magical and theosophic metaphysics like Burroughs and Gysin.

Whilst the culture and pompous rites of the Collège de Pataphysique with its “satraps” and “magnificences” parodically reworked the association of science and metaphysics, the parody of Swift’s academy of Lagado was practically made a live performance in the Oulipo. It operates on the grounds of the Swiftian assumption that poetic combinatorics are a fancy, but therefore have an artistic-fantastic potential to be exploited and explored. At the same time, the formalist constraints create a program for channeling artistic subjectivity, as opposed to—for example—the boundless romantic subjectivism of the anti-formal, but pointless Lettrist sound poetry of Isidore Isou and Maurice Lemaître from which the Situationist group emerged, too.

As opposed to Bense’s school with its rigorous constructivist modernism, Oulipo considered formalisms and computations neither a poetic end in themselves, nor a philosophical-ideological base. Oulipo created a computational poetics as anti-computational poetics, using computational formalisms for the sole end of circumventing them artistically. While Bense fought against semantics and imagination, Oulipo made it their game to let imagination fight against self-imposed formalism and have it triumph in the end.

Still, Oulipo’s poetics did not really reflect computations as cultural, and formalisms as loaded with meaning. Like others, it conceives of culture, imagination and meaning as something foreign to and struggling against formal rules. The slogan of 1990s digital art collective I/O/D, “software is mind control, get some” is anticipated in Oulipo only in its first half. Mind control is not embraced in Oulipian poetics, but overcome in a factually romanticist move. It comes as little surprise that Oulipo largely gave up on formal-algorithmic methods after spinning off a computer programming division *ALAMO* (“*Atelier de Littérature Assisté par la Mathématique et l’ordinateur*,”

⁵⁰ Georges Perec. *A Void*. HarperCollins, New York, 1994 1969. [75]

“Laboratory of Mathematically and Computer-Assisted Literature”) in 1973 and the deaths of Raymond Queneau in 1977 and Georges Perec in 1982. Nowadays, Oulipo focuses on improvisational, non-computational games like the writing of poems in between two subway stops.

Italo Calvino’s speculation on literature as computation also reflects upon Oulipo in which he had been a regular member. *Cybernetics and Ghosts* documents his exposure both to the structuralism of Tel Quel and Oulipian poetics, synthesizing the former’s theoretical, analytic understanding of language as a combinatorial system with the latter’s practical, synthetic poetics.⁵¹ In a later contribution to Oulipo proceedings, Calvino refers to his combinatorial prose composition as “anticombinatorics.”⁵² He distances himself from his earlier theory that the poet could be replaced by a machine, and adopts the Oulipo angle of the algorithm as a self-imposed constraint. With his own personal background as a neorealist and fantastic novelist who heavily drew from folk tales, his late prose works like *Invisible Cities*, *The Castle of Crossed Destinies* and “If on a Winter’s Night a Traveller” combine rigorous Oulipian composition—comparable to the novels of Georges Perec—with a fantastic imagination of space rendering them (and in particular the novel *Invisible Cities*) another para-manifestation of psychogeographical computing.

Abraham M. Moles’ computational aesthetics

Just as Oulipo’s anti-formalist thrust was less clear when the group started off as a “seminar of experimental literature,” its literary work was initially perceived as computational experimental literature. In 1962, physicist and philosopher Abraham M. Moles wrote a *first manifesto of permutational art (erstes manifest der permutationellen kunst)*⁵³ which was published by Max Bense in Stuttgart. Later, in 1971, he expanded it into a book *Art et Ordinateur (Art and Computer)*.⁵⁴ The “manifesto” combines structuralist and cybernetic theory with examples of concrete poetry, Oulipo works and pre-modern

⁵¹Italo Calvino. *Cybernetics and Ghosts*. In *The Uses of Literature*, pages 3–27. Harcourt, San Diego, 1982 (1967). [22]

⁵²Italo Calvino. *Prose and anticombinatorics*. In: Warren F. Motte, editor. *Oulipo. A Primer of Potential Literature*. University of Nebraska Press, Lincoln, London, 1986., pages 143–152. [23]

⁵³Abraham A. Moles. *erstes manifest der permutationellen kunst*. Stuttgart, 1963. [68]

⁵⁴Abraham A. Moles. *Art et Ordinateur*. Casterman, Paris, 1981 (1971). [69]

Lullist and proteic literature. In Moles' definition, permutational art is—unlike the Oulipian concepts of naive artisanship and self-imposed constraints—“experimental to the highest degree,” striving to “narrow down and exhaust the field of possibilities accessible through a set of rules.” Along with Bense and his information aesthetics, he even conceives of perception as something that can be formally described through technical information theory and functions as a reverse combinatory process.

In a seeming rehash of 17th century encyclopedic Lullism, Moles proposes to refound the arts on the basis of permutational combinatorics. The disciplines his manifesto covers in brief individual sections are mathematics, music, literature and poetry, mysticism, erotica and painting. These disciplines are, as in Leibniz' and Novalis' mathesis, reunited on the grounds of mathematical combinatorics. In tune with Bense, Moles writes both a poetics and an aesthetics in which a near-infinity of generative output likewise corresponds to a computational decoding of aesthetic phenomena. It is a totalism that includes both synthetic creation and analytical perception. In the attempt to map all arts and thinking onto algorithmic processes, Moles' theory resembles artificial intelligence research and its project to formally describe semantics as a higher complexity syntax. Moles' program appears not as rigorously totalist, however, because of its sketchy and highly speculative elaboration in this brief manifesto. In tune with Bense's philosophy and its grounding of aesthetics on technical information theory, it conceives of “permutational art” as a “fundamentally anti-semantic activity.” The manifesto is a historical document of cybernetics and its attempt at a universal science of technology that investigates human-machine interaction. Cybernetics largely faded out and was often considered obsolete in the 1970s until many of its ideas—most importantly the description of cultural processes in terms of technical processes—resurfaced in the 1990s, in the guise of technical media theory.

Source code poetry

One year after Moles' manifesto, the 9th issue of the Situationist International journal ran a polemic against Abraham Moles. Guy Debord calls him upon to “médite sur la valeur anti-combinatoire du mot,” “ponder the anti-combinatory value of a word.”⁵⁵ The phrase is reminiscent of Oulipo poetics and Calvino's “anti-combinatorics.” Such

⁵⁵Internationale Situationniste, editor. *Internationale situationniste. Édition augmentée*. Librairie Arthème Fayard, Paris, 1997 (1958-1969). [93], p. 411

overlaps actually existed. Noël Arnaud, member of the Collège de Pataphysique since 1952 and co-founder of the Oulipo, was involved also in the pre-situationist *Cobra* painters group and later, after the schism between Northern European Situationists and Debord's circle, became co-editor with Jacqueline de Jong of the *Situationist Times*. In 1968, Arnaud published a book *Algol*. The poems it contains are based on the vocabulary of the Algol programming language translated into French. The idea of Algol poetry came from Le Lionnais' first Oulipo manifesto from 1962. Still in the spirit of "experimental literature," it proposed "forays [...] notably into the area of special vocabulary (crows, foxes, dolphins; Algol computer language, etc.)."⁵⁶

For the first time, and unlike in classical computer-generated poetry like that of Theo Lutz and Brion Gysin, the programming language source code was acknowledged as having a poetic quality of its own. Arnaud's Algol source code no longer generated poetry, but was the poem itself. Program code ceased to be seen merely as a "transparent" technical tool detached from the perceived art work, but as an aesthetic object itself. For the first time, artistic programming was not a means to another end. Computer programming languages appealed to the Oulipo as yet another formal constraint. In comparison to standard human languages, all programming languages are drastically limited in their vocabulary and syntax, Algol even more than others. A "poor" language that limited a writer's freedom, Algol was yet another playground for Oulipo poets to overcome artificial constraints through imagination and cleverness.

This poetics highly resembles a "hacker" approach. Hacker culture, which—according to its mainstream historicizations and legends—originated at MIT in practically the same years as the Oulipo,⁵⁷ had the very similar goal of appropriating technology in creative ways which could, if necessary, include clever circumvention of superimposed barriers and limitations. So it is perhaps not surprising that hacker culture re-invented the genre of programming language poetry, but without knowing of its Oulipo precursors. In an internet newsgroup posting of 1991, Larry Wall, creator of the Perl programming language (itself a "hack" in its syntactical mixture of older Unix scripting languages), left the message

⁵⁶François Le Lionnais. Lipo: First manifesto. In: Warren F. Motte, editor. *Oulipo. A Primer of Potential Literature*. University of Nebraska Press, Lincoln, London, 1986., pages 26–28. [61], p. 27

⁵⁷Steven Levy. *Hackers*. Project Gutenberg, Champaign, IL, 1986 (1984). [63]

```
Print STDOUT q
Just another Perl hacker
Unless $pring
```

... which is proper executable Perl code. After his example, a number of Perl programmers wrote source code poems in Perl. They were collected, still in the same year, in Sharon Hopkins' paper *Camels and Needles: Computer Poetry Meets the Perl Programming Language*.⁵⁸ All the poems included in the volume are "artisanship" as Oulipo had advocated it. However, they are naive, non-ironic artisanship, being quite conventional and stereotypical poems about love or nature, only in the form of Perl source code or, the most cases, pseudo-code that looks like Perl but can't be machine-executed. It is poetry as a popular social game similar to limericks or Haikus.

Jodi

Source code poetry was reinvented for a third time in the mid-1990s net.art of Jodi. Jodi stands for Joan Heemskerk and Dirk Paesmans, a Dutch-Belgian artist couple. They were part of a network of artists who radically redefined digital art from so-called "interactive" high-tech graphical simulations to ironic low-tech works that played with bugs, incompatibilities and disruptions in software. Contrary to a slick, visually immersive digital art which treated the computer as a black box, Jodi aestheticize computers as self-immersed, often absurd generators of contingent data streams. The work *OSS*, for example, makes small browser windows which evade manual control pop up and fly about, forcing the user to take down the computer in order to regain control. The *OSS* desktop expands the misbehaving, uncontrollable user interface onto the entire Windows desktop. Yet Jodi's art does not tell an imaginary truth underneath the surfaces of software user interfaces. Instead, it exposes the surrealism of formalisms. A newer Jodi work for example simply uses a commercial car driving computer game, and lets the car make infinite, lawnmower-like circles, with high speed and squeaking tires, in the front garden of an American suburban family home. On <http://www.jodi.org> visitors are confronted with code in the form of cryptic error messages, blinking machine symbols and contingent tabular listings of numbers. It is code which often simply refers to other code—an error message for example linking to another error message linking to yet another error message. But moreover it is code which is not what it

⁵⁸Sharon Hopkins. *Camels and Needles : Computer Poetry meets the Perl Programming Language*, 1991. www.wall.org/~sharon/plpaper.ps. [47]


```

1 21:48:24 +0100 (MMi6 2001 (MMi6 2001 21:48:24 +0100 (MMi6 200
  1 21:48:24 +0100 (MMi6 2001 (MMi6 2001 21:48:24 +0100 (MMi6 200
    1 21:48:24 +0100 (MMi6 2001 (MMi6 2001 21:48:24 +0100 (MMi6 200
      1 21:48:24 +0100 (MMi6 2001 (Mon the discussions between
Mrinmoy / Shubhodip (bluwregraseqr werMi6 2001 21:48:24 +0100 (MMi6 200
  1 21:48:24 +0100 (MMi6 2001 (MMi6 2001 21:48:24 +0100 (MMi6
200      1 21:48:24 +0100 (MMi6 2001 (MMi6 2001 21:48:24 +0100
(MMi6 200      1 21:48:24 +0100 (MMi6 2001 (MMi6 2001 21:48:24

```

[...]

The above message appeared not in the E-Mail message body, but entirely in the subject line, trying to provoke errors, crashes and confusion in the distribution and display of the message. With such interventions, Jodi tried to refashion net cultural discussion forums from theoretical discussion forums about art and culture into practical artistic playgrounds. At the same time, their E-Mail art once more reduced the technical complexity of digital art, in sharp contrast to high tech museum installations. With their conflation of English everyday language, code fragments from programming languages, character encodings, markup languages and network protocol code, these messages furthermore created uncertainty and paranoia among naive readers who might think that the signs were malicious and contained viral code. Consequently, the Jodi website was temporarily shut down because its network provider believed that OSS was a computer virus. This was the practical proof that Jodi's artistic simulations of code in the medium of code were efficacious. (More on on p. 112.)

As imagined viruses, the codes raised doubt about their origins. All the more when a growing number of pseudonymous entities began sending out similar messages to mailing lists. Media theoretician McKenzie Wark characterized those message as follows:

This might be mangled machine English, or perhaps an English written by a machine programmed by someone who speaks English as a second language, or someone producing a simulation of some such.⁵⁹

It also remained unclear to which degrees computation had been at work in composing the messages. Just like it was doubtful whether their symbols were plain English or machine code, it was doubtful whether their originators were humans, computer programs, computer programs filtering and modifying human language, or the opposite, humans filtering the output of computer programs. McKenzie

⁵⁹McKenzie Wark. Essay: Codework. *American Book Review*, 22(6):1–5, September 2001. [100]

Wark and artist Alan Sondheim later coined the term “codeworks” for this kind of net.art.⁶⁰ With its uncertain origins and bizarre human-machine involvement, experimentation with artistic identity and subjectivity remained its core characteristic.

1337 speech

Codeworks reflect the uncanny underbelly of network communication in an age where the Internet is accessed largely by graphical browser and client programs, but with the constant awareness that non-graphical codes are running underneath the system. In mainstream computing, these codes revealed themselves only in error messages or as “blue screens of death” when the operating system crashes. Computer hackers had written source code poems and typograms with ASCII characters out of the aesthetic limitation of non-graphical terminal and command line computing between the 1960s and early 1990s. For the net.artists who worked in the age of the web browser, these non-graphical constraints were voluntary and self-imposed, very much like Oulipo’s constraints. Since ASCII typograms were hacker circumventions of technical limitations, they had an aura of subversion, and were hybridized with slang.

A particular slang developed in the subculture of “crackers,” hackers who break into network computers or crack the copy-protecting schemes of computer games. It replaces letters with numbers, for example “e” with “3” and “t” with “7” (because of their typographic similarity), so that a “1337 [leet] hax0r” becomes code speech for “elite hacker.” These substitutions originated in about the same time as similar codes in hip hop culture where, for example, “2pac” stood for rapper Tupac Shakur. Writing in 1337 slang, among others on websites defaced by crackers, became digital graffiti. Since the letter substitutions can be automated, there also exists simple text filter software—like the Unix filter “b1ff”—which transforms standard English writing into *733t speech*:

SINCE THE LETTUR SUBSTITUSHUNZ CAN BE
AU2M8D. THURE ALSO EXISTZ SIMPLE TEXT F1LTUR
SOFTWARE—LIKE THE THE MANEFRA1M O/Z
F1LTUR B1FF—WH1CH TRANS4MZ STANDARD
ENGLISH WR1T1NG IN2 733T SPEECH.

Clearly, the cracker slang and graffiti were a source of inspiration for net artists like Jodi. When net cultural mailing lists started to

⁶⁰Alan Sondheim. Introduction: Codework. *American Book Review*, 22(6):1–4, September 2001. [95]

filter out their disruptive codework around 1997, Jodi co-founded a mailing list *7-11* which had no editorial restraints of artistic E-Mail experimentation, and even let through all commercial spam. On this and other lists, artists like mez (Mary Anne Breeze) and Alan Sondheim transformed the former disruption aesthetic into new hybrid computer-English poetic vocabularies and languages. The result were works like the “Exe.cut[up]able statements” described in the very beginning of this paper.

Codework

In the early 1970s, Alan Sondheim began to work with programming code as material in the larger context of conceptual art. He experimented with Unix command line code in experimental writing as early as in the 1980s, and collaborated with younger net.artists since the 1990s. One of his codeworks reads as follows:

```
From: Alan Sondheim <sondheim@panix.com>
To: _arc.hive_@lm.va.com.au
Date: Thu, 9 Jan 2003 17:17:20 -0500 (EST)
```

sleeping and running zombies through bodies

```
CPU states: 4.7% user, 5.8% system, 0.0% nice, 89.4% idle:36 processes:
35 sleeping, 1 running, 0 zombie, 0 stopped:lm 4:20pm up 8 min, 1 user,
load average: 0.54, 0.26, 0.11: :Mem: 38664K av, 35084K used, 3580K free,
14956K shrd, 15080K buff Write vaginas through my CPU states: 4.7% user,
5.8% system, 0.0% nice, 89.4% idle! CPU states: 4.7% user, 5.8% system,
0.0% nice, 89.4% idle:36 processes: 35 sleeping, 1 running, 0 zombie, 0
stopped:lm 4:20pm up 8 min, 1 user, load average: 0.54, 0.26, 0.11: :Mem:
38664K av, 35084K used, 3580K free, 14956K shrd, 15080K buff Write vaginas
through my CPU states: 4.7% user, 5.8% system, 0.0% nice, 89.4% idle!
load average: 0.54, 0.26, 0.11: :Mem: 38664K av, 35084K used, 3580K
free,:35 sleeping, 1 running, 0 zombie, 0 stopped:lm 4:20pm up 8 min, 1
user,:CPU states: 4.7% user, 5.8% system, 0.0% nice, 89.4% idle:36
processes::Write vaginas through my CPU states: 4.7% user, 5.8% system,
0.0% nice,:89.4% idle! CPU states: 4.7% user, 5.8% system, 0.0% nice,
89.4% idle:36 processes: is sufficiently well-inscribed. - I consider the
following again, your CPU states: 4.7% user, 5.8% system, 0.0% nice,
89.4% idle:36 processes: ... enunciation inscribes me upon your token! CPU
states: 4.7% user, 5.8% system, 0.0% nice, 89.4% idle:36 processes:, load
average: 0.54, 0.26, 0.11: :Mem: 38664K av, 35084K used, 3580K free,
remembers my chisel My load average: 0.54, 0.26, 0.11: :Mem: 38664K av,
35084K used, 3580K is your language... load average: 0.54, 0.26, 0.11:
:Mem: 38664K av, 35084K used, 3580K free, calls forth births inscription,
hungered, making things. upon the time, load average: 0.54, 0.26, 0.11:
:Mem: 38664K av, 35084K used, 3580K free, is here, 00], 35 sleeping, 1
running, 0 zombie, 0 stopped:lm 4:20pm up 8 min, 1 user,? ... inscription
```

```

is stopped:lm 4:20pm up 8 min, 1 user, load average: 0.54, 0.26, 0.11:
:Mem: on black stone, it's inscription? Are you satisfied with your load
average: 0.54, 0.26, 0.11: :Mem: 38664K av, 35084K used, 3580K free,?
load average: 0.54, 0.26, 0.11: :Mem: 38664K av, 35084K used, 3580K free,
1086 is the perfect proclamation. CPU states: 4.7% user, 5.8% system,
0.0% nice, 89.4% idle:36 processes: 35 sleeping, 1 running, 0 zombie, 0
stopped:lm 4:20pm up 8 min, 1 user, load average: 0.54, 0.26, 0.11: :Mem:
38664K av, 35084K used, 3580K free, 14956K shrd, 15080K buff Write vaginas
through my CPU states: 4.7% user, 5.8% system, 0.0% nice, 89.4% idle!
load average: 0.54, 0.26, 0.11: :Mem: 38664K av, 35084K used, 3580K
free,:35 sleeping, 1 running, 0 zombie, 0 stopped:lm 4:20pm up 8 min, 1
user,:CPU states: 4.7% user, 5.8% system, 0.0% nice, 89.4% idle:36
processes::Write vaginas through my CPU states: 4.7% user, 5.8% system,
0.0% nice,:89.4% idle! load average: 0.54, 0.26, 0.11: :Mem: 38664K av,
35084K used, 3580K free,:35 sleeping, 1 running, 0 zombie, 0 stopped:lm
4:20pm up 8 min, 1 user,:CPU states: 4.7% user, 5.8% system, 0.0% nice,
89.4% idle:36 processes::free,:35 sleeping, 1 running, 0 zombie, 0
stopped:lm 4:20pm up 8 min, 1:38664K av, 35084K used, 3580K free, 14956K
shrd, 15080K buff Write vaginas Your enunciation names my stopped:lm
4:20pm up 8 min, 1 user, load average: 0.54, 0.26, 0.11: :Mem: !

```

The work is based on the output of the Unix system command “top” which displays a list of running processes, memory and central processor load. “Zombie” is a technical Unix term for a program process that can no longer be terminated with the “kill” command. Sondheim’s text takes these descriptors—or “semantics,” as computer science would call it—literally. He reads the output of the program as a physical inscription of bodies, as performance art and a subjective utterance in the medium of computer software. It gets reformatted and partially rewritten, in an operation that blurs the boundaries of machine and human, syntax and semantics, with words and phrases mapping bodies and sexuality. Subject and object, syntax and semantics, formalism and culture become inseparably entangled, crisscrossing and writing over each other.

This is not simply a poetic metaphorization because the technical apparatus of writing becomes a part of the text. There is a feedback of textual input, output and processing inside the text and within the medium of code. This conflation of source code, data and processing could be called recursive, since recursion means that a process processes itself. Yet it is not a formal-mathematically “clean” recursion. The text is not a result of a pure computation, but involves human editing, rendering the recursion a simulation, rhetorical, reflexive.

The “codeworks” by Jodi, mez, Alan Sondheim and other artists manifest a most radical understanding of formalisms as meaningful. That their codeworks rarely execute properly, being imaginary code and confluences of machine code with human conversational

language, reflects this understanding. The codeworks appropriate languages that were designed to be asemantic—programming languages, protocol code, shell commands—to unveil and elaborate their metaphorical and physical inscriptions, implications, and engendered meanings lurking between the lines. In other words, they do not just reflect words made flesh, but words and codes being flesh.

CHAPTER 4

Automatisms and Their Constraints

Artificial Intelligence

From magic to codeworks, three cultural modes of computation have evolved:

(1) Instruction code as synthesis.

A small amount of code generates, from a fixed set of data inscribed into that code, a large body of output. The principle applies to all combinatorics from Lull to generative art. In this kind of computation, the code itself is exempt from the algorithmic process. It triggers the process, but is not being processed itself. The result of the synthetic process is often imagined to be infinite, total and exhaustive.

(2) Instruction code as analysis.

Either

(a) In a reciprocal operation to synthetic combinatorics, a large body of information is being reduced to a smaller amount of source code. Prominent examples are the kabbalistic condensation of the Torah to the letters JHWE and the reduction of language to a combinatorial set or transformational grammar in structuralist and Chomskyan linguistics.

(b) Alternatively, the code does not synthesize a fixed set of data into a whole, but takes apart arbitrary input. This the principle of Tzara's Dadaist poem, cut-ups and Markov chains. The resulting work is not conceived as exhaustive or total, but contingent and collage-like. The instruction code however exists outside that contingency. It has to remain intact, and not take itself apart, in order to be operational. The strict separation of static instructions and contingent data contradicts the assumption of a "chance operation." This is the paradox of all aleatory art, including concrete poetry and the music of John Cage.

(3) Self-processing instruction code

- (a) as formal recursion; the instruction goes into strange loops by processing itself. It modulates or modifies itself in that process. This is a standard procedure in programming languages such as Lisp. The classical Greek paradox of the lying Cretan (“all Cretans are liars. I am a Cretan”) is the oldest known example of a logical recursion in language.
- (b) as informal conflation.

Unlike formal recursions, codework processes instruction codes semantically since it understands formalisms as meaningful, loaded with metaphors, physical and subjective inscriptions. The historical precondition of this poetics is an understanding, and availability, of computational instruction codes as material. Code is no longer synthesized and created from scratch in clean-room laboratories, but abundantly flows through computers and networks. It is understood as unclean, messy, strangely meaningful, a trigger not only of mathematical processes, but imagination and phantasms.

These imaginations and phantasms exist in the synthetic and analytical poetics of computational instruction code as well, but it is either—as in Lullism for example—not reflected upon as imaginative, or is so reflected outside the medium of computational code itself, such as in the descriptive realist prose of Swift and Borges.

Except for synthetic combinatorics, the question implicit in the above approaches is no longer whether algorithms can replace human creativity and cognition. Instead, they reflect computation as a cultural phenomenon of its own. The connection between computations and human subjectivity, imagination, politics and economy is therefore intricate, contradictory and rich with metaphors. That computation could simply replace human creation by mapping human cognition one-to-one onto algorithms appears as naive, or only an idea about the cultural impact of computation. Nevertheless, the idea did not die out with encyclopedic Lullism and its Swiftian satire, but was reinvented in the 20th century as artificial intelligence. In linguistic-cognitive terms, the project of A.I. research is to prove that such a thing as “semantics” does not exist and is just a higher-order syntax. So far, A.I. has failed to deliver the practical proof. Instead, outside of its stated goal it has produced interesting technological and cultural

by-products for fifty years, such as the programming language Lisp, or the GNU project that was initiated in the MIT artificial intelligence lab.

Athanasius Kircher's box

The idea of automating language, art and cognitive reasoning existed already in 17th century Lullism. Its computational device, both in the form of algorithms and hardware, were orders of magnitude more primitive. In 1674, three years after his permutational sonnet, Quirinus Kuhlmann published his correspondence with Athanasius Kircher in a book *epistolae duae*.¹ It documents an early debate about automatically generated art and its cognitive limitations. In his letters, Kuhlmann rejects a purely technical application of Lull's combinatorics, claiming that "knowing Lullus does not mean to have knowledge in the *alphabetum* of his *Ars* in order to build syllogisms with it, but to grasp the true power from the universal book of nature that is hidden underneath it, and apply it to everything."

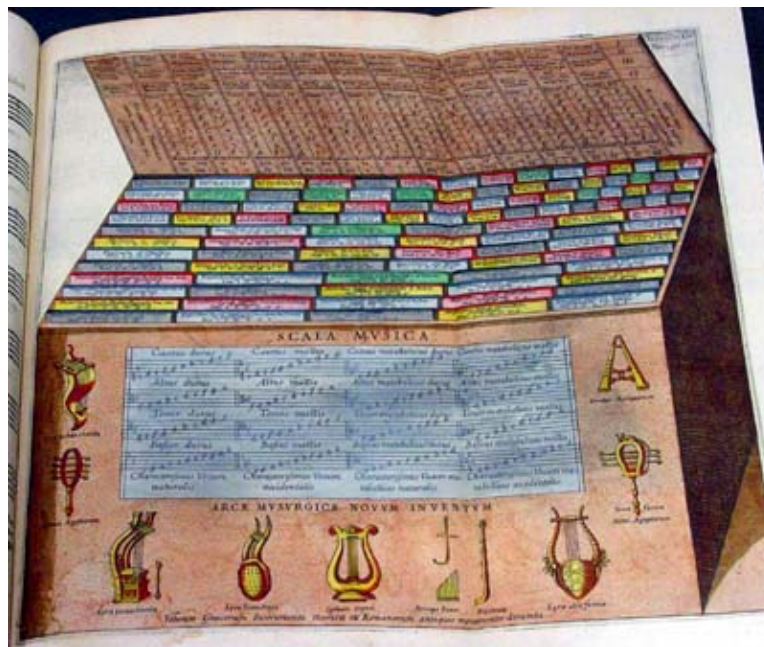


FIGURE 1. Athanasius Kircher's design of a music composition machine

¹Quirinus Kuhlmann. *Epistolae duae*. Lotho de Haes, Amsterdam, 1674. [58]

Kircher rejects this viewpoint on theological grounds, warning Kuhlmann in drastic words that he might cross the line of heresy. His own application of Lullism, Kircher writes, forays more into technical realms, including a music generation machine; a device described also in his book *Musurgia Universalis* and of which a graphical reproduction exists. Along with the musicology of early 17th century Lullist Marin Mersenne, Kircher's *Musurgia* is often cited as a precursor of contemporary computer-generated music.

Kuhlmann's insistence on a theosophical "book of nature" underneath Lullist computation might seem like traditional metaphysical thinking, but his critique of automatically generated arts is hands-on: He rejects the idea of a machine, or "box" ("*cista*") as Kircher calls it, that generates poetry, arguing that such a machine could indeed be built, but it would not produce good artistic results. One could teach, Kuhlmann writes, every little boy verse composition through simple formal rules and tables of elements ("*paucis tabellis*"). The result however would be versifications, not poetry ("*sed versûs, non poëma*"). Unlike artists before and after him, Kuhlmann does not consider algorithmic composition and artistic subjectivity contradictory. He reconciled what he saw as the macrocosmic order in computation and his own microcosmic artistic subjectivity by eventually fashioning himself into a prophet and messiah. This way, he took the concept of the artistic genius to its ultimate extreme even before it was nominally invented in the 18th century. The rhetorical "ingenium" as it had been laid out in 17th century rhetoric from Sarbiewski to Morhof was a pretext of this development. With its odd condition of being at once (a) technical and (b) human subjective wit, it later branched into "genius" and "engineer" and paved the way for all controversies over art and the extent to which it could be formalized and automated. It is an issue which applies not only to art, but to any meaningful, semantic phenomenon, including language and culture. The debate between Kuhlmann and Kircher therefore was a first controversy over artificial intelligence, and the potentials and limitations of machine cognition.

John Searle's Chinese Room

Three hundred years later, in 1980, Kircher's and Kuhlmann's debate was partially rehashed in a dispute between language philosopher John R. Searle and various artificial intelligence researchers. In his

paper *Mind, brains, and programs*, Searle sketches a thought experiment of artificial language cognition.² A person who does not understand Chinese is enabled to communicate in Chinese solely by formal instruction. The person sits in a closed room and is given questions written in Chinese characters from outside (figure 2). He transforms the Chinese characters into other Chinese characters solely through the completely formal, step-by-step guidance of an English rule book. Thanks to the precise transformation rules, the results are flawless Chinese replies to the Chinese questions. Searle argues that the person, although people outside would think it spoke Chinese perfectly, would not understand the language at all.



FIGURE 2. Illustration of Searle's Chinese Room from <http://www.macrovu.com/CCTMap4ChineseRm.html>

When Searle's paper first appeared in the journal *The Behavioural and Brain Sciences*, it was accompanied by a number of critical responses and attempts at refutation by A.I. scholars. From a "hard" A.I. standpoint as Searle calls it, one could argue that human cognition is no more than higher-order formal computation and syntax manipulation, only that humans don't have an understanding and self-awareness of the formal processing in the brain. Another counter-argument is that the resulting whole consisting of the chamber, the inmate and the book should be considered intelligent, i.e. the entire system and not its single components such as the room's inmate. Perhaps

²John R. Searle. *Minds, brains, and programs*. *The Behavioral and Brain Sciences*, 3:417-456, 1980. [91]

the strongest critique is that Searle's argument is entirely ontological and metaphysical because it does not matter whether the system has an "understanding" as long as it perfectly acts as if it would understand. When the simulation of cognition is as good as cognition itself, it does not matter—or is just a metaphysical question—whether it's a simulation or not. This topic has been reflected in many Science Fiction novels, most popular is Philip K. Dick's *Do Androids Dream of Electric Sheep* from 1968 in its adaption as the Hollywood film *Blade Runner*.

Remarkably, both Searle and his A.I. research critics assume that something was technologically feasible which, in its time and still today, is science fiction. There simply exists no rule book—in other words: no algorithm—for transforming Chinese questions into Chinese answers. The proof that such a book could be written solely on the basis of mathematical computation, or formal logic, is still outstanding. Both A.I. critics and A.I. adherents assume in pure speculations that the technology is capable of what they describe.

The whole field of "artificial intelligence" is somewhat odd in that there exists no hard scientific notion of intelligence yet in the first place. So there is no clearly defined objective of the research. If intelligence, for example, is simply defined as logical-mathematical calculation, then of course computers are intelligent machines. If intelligence is defined through the ability of forming opinions, then a thermostat is "intelligent" having the opinion that a room is too cold or too hot (to use an original example from the Searle debate).

A.I. contradicts basic principles of modern empirical science. Research is supposed to be based on observation and heuristics, not superimposed notions or categories which are not even clearly defined themselves. Unlike empirical science, A.I. as exemplified by the Turing test, knows its results including the proof beforehand. It searches for a process that fits the predefined result and proof, forcing reality to fit the result instead of having the result fit reality. This makes A.I. research an heir of medieval scholastic science and brings up very similar philosophical issues to those of Lullism as criticized by Kuhlmann and Swift.

The debate over Searle's Chinese room reveals how the question of whether computational hardware and software is capable of cognition is completely ruled by a cultural imagination and phantasm—both on behalf of A.I. research and of Searle—of what mathematical computations might be capable of. The debate is like a hypothetical

argument over the environmental impact of perpetuum mobiles, instigated by perpetuum mobile research and its entirely speculative promises.

Georges Perec's *Maschine*

Commissioned in 1968 by Saarländischer Rundfunk in Germany, Oulipo poet Georges Perec wrote a radio play *Die Maschine* (*The Machine*), in collaboration with his German translator Eugen Helmlé. The play simulates, as the foreword explains, a computer “systematically analyzing and dissecting Johann Wolfgang von Goethe’s poem ‘The Wanderer’s Night Song’.” In the English translation contained in Perec’s script, Goethe’s short poem (a classic of German literature taught at every high school) reads as follows:

over every hill
is repose
in the trees, you feel
scarcely goes
the stir of a breeze.
hushed birds in the forest are nesting.
wait, you’ll be resting
soon too like these.³

Perec’s imaginary computer consists of three “memory” units and one “control” unit. In the radio play, they appear as different speakers (figure 3). The control unit commands the performance of certain computations on Goethe’s text, for example, on the structure of rhyme, the number of letters, or the average of letters per verse. It commands word by word readings, readings of groups of two words and further transformations of the text which implement practically all the poetic algorithms the Oulipo invented or reinstated from pre-modern poetics. The three memory units recite the different results of the text transformations according to the control unit’s instructions. Taking up eighty pages in the manuscript, the computations show a pataphysical machine at work. They render themselves absurd in creating more and more pointless variations of Goethe’s poem. In the end, the machine performs synonym lookups and international translations of “rest” into “silence” and “peace,” and exhausts itself in the process. The words uttered by the three memory units are also spoken by the control unit; the data overwrites the program code, and the system crashes. The semantics of Goethe’s poem, with the instruction to

³ibid., p. 46

“wait” and “find rest,” becomes the instruction code of the machine eventually. The text which originally should be input data turns into a program, overwriting the original program of the pataphysical machine. So it works just like an E-Mail virus. Such a virus, thought to be only data for transmittance, actually executes on a computer system as an algorithm, takes over control, and brings the infected system to a grinding halt. *Die Maschine* is a tribute to Goethe’s poem as a powerful text which makes all formal processing running amok and fail because its semantics resist syntactical processing.

Speicher 1	Speicher 2	Speicher 3	Kontrolle
aufnahmebereit	aufnahmebereit	aufnahmebereit	speicher in aufnahmebereitschaft
 o 			TON hier erato. um ihre frage direkt durchzuprogram- mieren, stecken sie bitte die perforierte karte in die lesespalte und drücken sie die tasten a und d KLICKEN TON titel des gedichts
wanderers nachtlid	6. september 1780	goethe johann wolfgang von, 1749-1832	entstehungsdatum autor des gedichts
	deutsch		originalsprache des ge- dichts
	über allen gipfeln ist ruh,		wortlaut des gedichts
	in allen wipfeln spürest du		

FIGURE 3. Manuscript of Perec’s radio play, with the text of the three memory units and the control unit in separate columns

Perec’s radio play is perhaps the first work of computational art reflecting the limitations of computers and aestheticizes computer crashes. It demonstrates, in a sometimes comic, sometimes painful formal exercise, how the pure syntactical processing of language by itself is poetically pointless, except as a cultural reflection of computation. It dramatizes, so-to-speak, the practical failure of Searle’s Chinese room because the algorithmic rulebook does not work as advertised. The poetics and artistic use of algorithmic processes therefore does not lie in simulated language composition, but in a cultural reflection of computations as computations, with all their formal limitations. This poetics was either not understood or not continued before net art and codeworks created an ironic digital art in the 1990s.

The algorithmic processing of Goethe's poem in Perec's radio play is in the end no different from the stochastic computer analysis of literature Perec's fellow Oulipian Italo Calvino ridicules in "If on a Winter's Night a Traveller," and the analytical methods of Bense's information aesthetics. According to Reinhard Döhl, an experimental poet, radio artist and member of Bense's group, Perec's *Die Maschine* was such a critical blow to the efforts of the Stuttgart group that, after listening to the radio play, it completely gave up computer poetry. The effect could only be so devastating because the group had conceived of computational or, to use Bense's term, "artificial" poetry and art not in ironic and cultural terms, but as a formalist-scientific foray into new realms of language. Perec showed that poetic algorithms weren't new and involved major limitations. In an essay on computer poetry, Döhl writes that Perec's radio play "seemed to us, as people who had gone from text to computing, like a preliminary end point. [...] We then did not follow these approaches anymore except in lectures and discussions, but extended our interest in artistic production with new media and notation systems into different directions."

Enzensberger's and Schmatz's / Czernin's poetic machines

In the 1970s, both Oulipo and concrete poetry had largely left computer experimentation behind or delegated it to specialist subprojects. At the same time, German poet Hans Magnus Enzensberger occupied himself with machine-generated poetry, as he wrote in 2000, out of boredom and frustration with the 1968 political movement having "dissolved into hangovers, sectarianism and violent fantasies."⁴ He retreated to "certain language and mind games which had the advantage of being obsessive."⁵ For months, he worked on the design of a poetry machine, "day and night," as he writes, like "hackers, gamblers who place their hopes in systems, and kids who are addicted to computer games."⁶ The result was a synthetic combinatory verse generation device similar to Queneau's *100,000 Billion poems* which Enzensberger knew and referred to in his 1974 project paper. He also references Harsdörffer, Kircher and Lull. The machine wasn't built until 2000, with a basic configuration of a PC computer program as a control unit and a mechanical letter display as used on airports as the letter display. Like Oulipo and concrete poetry, Enzensberger no

⁴Hans Magnus Enzensberger. *Einladung zu einem Poesie-Automaten*. Suhrkamp, Frankfurt/M., 2000. [34], p. 13

⁵ibid.

⁶ibid.

longer pursued computer-generated poetry after this first experiment and even considered leaving the design paper unpublished.

In 1990, the Austrian experimental poets Franz Josef Czernin and Ferdinand Schmatz commissioned the development of a computer program *POE* intended to serve as a computational toolkit for poets just like music composition and sound synthesis software served musical composers.⁷ *POE* differed from previous poetry programs in that it wasn't a poetry generator, but a piece of software for computer-aided poetry composition. It did not synthesize built-in words, but worked with any input, and provided a whole set of transformation algorithms to choose from instead of only one. As such, it very much resembled the command line userland of Unix with its multiple single-purpose text filtering tools like *grep*, *wc* and *sort*. Among the algorithms provided in *POE* were Markov chains and permutations. Czernin and Schmatz abandoned the project soon, however, drawing a similar conclusion to Percec; namely that the machine would not help to modify text according to semantic criteria or, in the case of *POE*, not even be able to perform grammatical transformations of input text. For Bense's Stuttgart group, for Enzensberger and for Czernin and Schmatz, poetic computing failed with the insight that its reality did not live up to artificial intelligence promises.

Software dystopia: Jodi

In its pataphysical subversion of artistic and scientific formalisms, Georges Percec's *Die Maschine* parodies computation and its ideological use as a weapon against an aesthetics of human subjectivity. The mere concept of anti-subjectivity through programming turned out to be driven by likewise subjective, personal agendas. That formalisms could be bizarre and eccentric was one of the aesthetic insights of net.art in the second half of the 1990s. Vuk Cosic, who coined the "net.art" name, credits Oulipo as one of his major influences.⁸ He and his fellow net.artists also drew heavily from hacker cultural artisanship such as ASCII art, program code poetry, graphics demos, games and viral code. Jodi's artistic exploration of the aesthetics of computer crashes, of source and protocol code, the contingency and absurdity of operating system and browser user interfaces, the aesthetics of computer games is computational art under

⁷Franz Josef Czernin and Ferdinand Schmatz. Notes about the Poetry Program *POE*, 1990. http://www.aec.at/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=8950. [27]

⁸Vuk Cosic. *Contemporary ASCII*. Kapelica, Ljubljana, 2000. [25], p. 14 and 32

the new conditions of ubiquitous personal and network computing. Jodi take apart the codes of everyday computer culture and modern art, combining the electronic graffiti aesthetics of hacker culture with pataphysical subversions of high modernism like those of the Oulipo.

While the crash of Perce's machine affected only a fictitious, partly metaphorical computer, Jodi let real computers crash, or at least pretended it via computer graphical *trompe l'oeil*. In Jodi's works, computing turns from a tool into an aesthetic end in itself. It is not a means of production like algorithms in generative art, and it does not even pretend to produce anything meaningful at all, in contrast to the way Perce's *Maschine* processed Goethe's poem. In Jodi's art, software, computations, code, user interfaces themselves are the medium of aesthetic reflection and play. The underlying concept of technology corresponds to that of Oulipian pataphysics since it reads computations as constraints, not expansions, of possibilities. Just like Perce's *Maschine* put an end to the utopian hopes of transgressing human limitations in early computer art, Jodi's art puts a sarcastic end to the utopias of audiovisual personal computing, from the graphical user interface as it was invented in the 1970s at Xerox PARC Labs to the utterly failed expectations of immersive three-dimensional "virtual reality" computing in the 1990s.

Unlike 1960s computing and generative art and its indebtedness to cybernetics, these newer developments and utopias had been driven by a McLuhanite concept of "media." The inventor of the GUI, Alan Kay credits McLuhan for the initial inspiration of his work, saying that it began with the insight that the computer was a "medium."⁹ In its most powerful manifestations, "media art" took the same theoretical base apart. Nam June Paik's early gutted-out TV sets were the critical counterpart to McLuhan's "global village," the work of Jodi and fellow net.artists the much-needed counterpoint to the high tech kitsch of "virtual reality."

Jodi's contingent, anti-usable website and software does not perpetuate utopian promises, but reflects failed promises as they manifest themselves in a daily life experience with software bugginess, instability, unreliability, user interface absurdity and other constant frustrations of "ease," "transparency" and "plug and play." But like the Oulipian constraints, Jodi's dystopian computing creates a paradoxical moment of freedom. The computer is no longer all-encompassing,

⁹Alan Kay. User Interface: A Personal View. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 191–207. Addison Wesley, Reading, Massachusetts, 1990. [54]

it no longer provides an imaginary screen of unlimited expressive possibilities. Instead it controls its users, turns them—as Jodi’s art reflects—into clicking slaves and prisoners of a maze of icons, menus and unintelligible code. By mapping this maze and tracing the limitations of the system, Jodi’s art allows playful human agency. “Interactive art” in Jodi’s work is not, as in classical “interactive art,” a behaviorist simulation of interactivity through a predefined set of actions and reactions. Instead, it is a true interactivity which encourages humans to interact with the system in unforeseen ways, in ontological and not stochastic indeterminacy—for example, by simply shutting off the machine or throwing it out of the window.

Software dystopia: Netochka Nezvanova

Code as cult. Jodi’s aesthetic of contingent codes and user interfaces has been contrarily adapted as outright user enslavement. Using a comparable aesthetic of contingent and unintelligible code, the antiorp / integer / Netochka Nezvanova project turned the notion of proprietary software to its ultimate extreme. Dubbed by online magazine Salon.com the “most feared woman on the Internet,”¹⁰ N.N. turned up in the mid-1990s on various net.art and electronic music-related mailing lists and bombed them with messages written in a private codework language:

```
Empire = body.

hensz nn - simply.SUPERIOR

per chansz auss! `reason` nn = regardz geert lovink + h!z !lk
az ultra outdatd + p!t!fl pre.90.z uezttern kap!tal!zt buffoonz

ent!tl!ng u korporat fasc!ztz = haz b!n 01 error ov zortz on m! part.
[ma!z ! = z!mpl! ador faz!on]
geert lovink + ekxtra 1 d!menz!onl kr!!!!ketz [e.g. dze ultra unevntfl \
borrrrrrr!ng andreas broeckmann. alex galloway etc]
= do not dze konzt!tuz!on pozez 2 komput dze teor!e much
elsz akt!vat 01 lf+ !nundaz!e.

jetzt ! = return 2 z!p!ng tea + !zolat!ng m! celllz 4rom ur funerl.

vr!!endl!.nn

1001 ventuze.nn
```

¹⁰Katharine Mieszkowski. The most feared woman on the internet. *Salon.com*, 2002. <http://www.salon.com/tech/feature/2002/03/01/netochka/>. [67]

```

/_/
      \      /
      \    /  i should like to be a human plant
      _{
      _{/
          \
          \_ \  i will shed leaves in the shade
                because i like stepping on bugs

```

The messages were obviously composed with the help of algorithmic filters, substituting for example the letter “i” with “!” or the word “and” with “+.” The project was radical chic not only in its syntax, but also in the semantic content of the messages that attacked well-known net cultural activists as “korporat fasc!ztz” (“corporate fascists”). The style much resembled that of cracker cultural *leet speech* (see p. 98). To obscure the origins of N.N.’s messages, a web of servers and domain registrations spanning New Zealand, Denmark and Italy was created. The messages pointed to likewise cryptic websites such as <http://www.m9ndfukc.org> and <http://www.eusocial.org>. After displaying codework writing similar to that on the mailing lists, the sites eventually lead to pages advertising *NATO.0+55*, an expensive video realtime processing plug-in for the musical composition program MAX. It is known today that N.N. was a collective international project, with the person who wrote *NATO* differing from the one who wrote the message quoted above. The project presented itself as a sectarian cult, with its software as the object of worship. In a wilful perversion of proprietary software licensing, *NATO* licenses were revoked if licensees critically commented upon Netochka Nezvanova in public. The business model was to let people buy into an underground and a cult. Digital artist Alexei Shulgin characterized N.N. as a corporation posing as an artist, reciprocal to artists who had posed as corporations before. Local cults of *NATO* VJs used N.N. style in their names and acronyms. Like a successful leader of a sect, the programmer of *NATO* eventually bought himself a Ferrari sports car.

Scientological mind programming. That the mind could be programmed like a computer was (and is) the basic idea of L. Ron Hubbard’s *Scientology* cult. It developed and continues to sell an occult system of systematic mind reprogramming. Its emblem, an ornamented cross, was derived from both the Rosicrucian cross (see p. 51) and Aleister Crowley’s OTO cross. Hubbard had been a member of a Rosicrucian organization and the OTO (*Ordo Templis Orientis*)

before he went on to found Scientology. Scientology was his refashioning of occult and magical techniques into, at least superficially, a “scientific” technology that used lie detectors and borrowed from popular science behaviorism, cybernetics and linguistics. A device which Scientology credits as a precursor of its “tech” is the “semantic differential” of fringe language philosopher Alfred Korzybski. Looking like an abstract string puppet, it was used as a meditative object and linguistic para-computer to teach students the radical dissociation of signs from things.

Both Crowley and Scientology made “total freedom” their main slogan. Scientology sought to achieve this freedom through methods that included a Korzybskian reprogramming (or deprogramming) of the meaning of words and word-permutational mind games. Enlightenment is gained in a gnostic hierarchy of gradual stages of reprogramming. Like other religions and belief systems, Scientology creates its own reality and involves capitalist commercialism in marketing its program.—A commercialism that has been rivalled only lately, but very successfully and with a similar clientele of pop cultural celebrities by the Kabbalah Centre. One could say that it counters Hubbard’s crypto-Kabbalah with a modernized and simplified version of the original Kabbalah.—Scientology pioneered the legal crackdown on the Internet when, as early as in 1995, it sued critics for copyright violation because they had published details of the Scientology course programs online. The idea of internal knowledge as “intellectual property,” a capitalist asset with infinite value propositions, was pre-empted by Hubbard himself when he ordered the continuous increase of prices for his writings. Living on a capitalist marketing of immaterial goods whose value was secured through copyright, Scientology is the oldest proprietary software company of the world. It showed that a profitable business could be founded on selling programs. Scientology actually converged with the proprietary computer software world when Microsoft decided to integrate *Diskeeper*, a PC program written by the Scientology company *Executive Software*, into its Windows 2000 operating system.

The similarity of Netochka Nezvanova’s aesthetics, the “m9ndfuke” and of her business model to Scientology might not be accidental. One of the key players in the N.N. project, Andrew McKenzie, makes numerous references to Scientology in the work he is better known for, his industrial music project *Hafler Trio*. One of their tracks is called *the sea org* which was the name of L. Ron Hubbard’s “cadet” organization on the ship where he lived, cruising international waters to circumvent arrest warrants. The Hafler Trio slogan “wash your brain

think again” strongly resembles the Scientology program of “clearing” one’s brain through erasing “engrams,” traumatic memories.¹¹ William S. Burroughs, the main source of inspiration of the industrial music movement, too stands for the role Scientology and its concept of the mind as a reprogrammable behavioral machine played in the occult underground history of 20th century software and computational arts. (Other followers of Hubbard’s ideas were, temporarily, John Cage and Morton Feldman.)

In *The Electronic Revolution*, Burroughs writes: “Ron Hubbard, founder of Scientology, says that certain words and word combinations can produce serious illnesses and mental disturbances.” While commenting on the theory with some scepticism, he goes on speculating about Hubbard’s “engrams” and “reactive mind.” He sketches a scenario of a cut-up movie that reads as if it were directly taken from a Scientology session:

Here are some sample RM [“reactive mind”] screen effects . . .

As the theatre darkens a bright light appears on the left side of the screen. The screen lights up.

To be nobody . . . On screen shadow of ladder and soldier incinerated by the Hiroshima blast

To be everybody . . . Street crowds, riots, panics

To be me . . . A beautiful girl and a handsome young man point to selves . . . To be you . . . They point to audience

Burroughs experimentally accepts Scientology concepts because they cater to his idea that language is a viral code yielding immediate physical effects. After all, Hubbard’s ideas had the same magical and occult sources as his own underground computational linguistics.

In this thinking, “total freedom” dialectically coincides with total control. The I/O/D slogan of software as “mind control” plays with the same dialectic. When I/O/D’s experimental *Web Stalker* browser removes interface abstraction, typographical sugar-coating and smoothness and unveils the underlying code—including HTML

¹¹In an interview with John Duncan, McKenzie says: “This is one of the good things about Scientology—they’ve got this big list of the things which are used subliminally, they say, in control techniques to subdue people because it tells you something that you must be anyway, you can’t help being, to be a body. To be here. To be now. And of course you hear this and go ‘fuck!’ And they have this thing about having simultaneous commands, stand up / sit down. Well you can’t do them both at once, so what they call your ‘reactive mind’ gets completely confused.” <http://www.johnduncan.org/stop.html>

source code and http protocol communication—, it frees the cultural technique and imagination of web browsing from its conventional metaphors.¹² At the same time, it maps the World Wide Web as a tightly ordered and controlled space. Freedom and mind control, in the end, don't contradict each other.

From dystopia to new subjectivity

If there is a common denominator of such diverse net.artists as Jodi, I/O/D and Netochka Nezvanova, it could be called the dystopia of computer software. Its manifestations however are diverse: playful-anarchic dystopias of Jodi's browser, desktop and game software; political-analytical dystopias of the Web Stalker; occult-corporate dystopias of Netchoka Nezvanova's operation *m9ndfukc*. To think of computation in dystopian, not naive utopian terms is what makes a critical and analytic approach to software possible in the first place. It implies thinking outside the system, and allows appropriation of computation as artistic material and subjective expression. The poetic codeworks of mez rephrase computer utopias on net.art's dystopian grounds, imagining a fusion of human bodies and machines in the medium of code. It is a "virtual reality" and "cyberspace" art that, unlike its techno-naive forerunners, is iconoclastic. Having given up on real computations and immersive imagery, it creates imaginary, impure computations, for example in the text cited at the beginning of the second chapter:

```
N.terr.ing the net.wurk---
::du n.OT enter _here_ with fal[low]se genera.tiffs + pathways poking
va.Kant [c]littoral tomb[+age].
::re.peat[bogging] + b d.[on the l]am.ned.
::yr p[non-E-]lastic hollow play.jar.[*]istic[tock] met[riculation.s]hods
sit badly in yr vetoed m[-c]outh.
```

These codes create, in the original sense of the word, science fiction. They generate a "new flesh" as it is imagined in David Cronenberg's film *Videodrome*, a melting of technology and human bodies. In this piece, this fusion happens purely in the medium of the imagination of the subject "N.terr.ing the net.wurk." Mez's text is, at it seems, based on a chat server login message that warns people not to enter with false identities. She transcodes it into "du n.OT enter _here_ with fal[low]se genera.tiffs + pathways poking va.Kant [c]littoral tomb[+age]." The old science fiction of computing as cultural and epistemological disturbance morphs into computing as a

¹²I/O/D Web Stalker homepage: <http://www.backspace.org/iod/iod4.html>

fiction. The code in this fiction is sexual as it is attached to the subjectivity of the person who interacts with it. Computer and network codes accumulate into personal diaries, and build cyborgs in the imagination. One could call mez's codeworks a fantastic realism of the Internet.

This realism is part of a larger network of a post-constructivist, post-clean room computational art that had been instigated by net.art. In this art, software has turned from a purified machine process into something embedded into cultural codes: version numbers, updates, protocols, interfaces, visible and invisible codes and cultural conventions of network communication.

CHAPTER 5

What Is Software?

A cultural definition

The previous chapters have spoken of “computations,” meaning calculation and algorithms performed either in the medium of language itself or with the help of mechanical or electronic devices. For many speculative computations, like those in the *Sefer Yetzirah* with its obscure mechanical device, it’s difficult, if not impossible, to draw a clear line between symbolic-imaginary and material processing, software and hardware.

What is software? Simply defined as algorithms, software would fail, first of all, to encompass the vast speculative imagination from magic to codework in which computational code rarely is pure algorithms, but a metaphorical hybrid. A strictly formal definition of software would also fail to describe the line of the Steven Seagal movie, “300,000 pages of code. Or 60 minutes of triple-X rubber-and-leather interactive bondage porno.” That line shows that such a cultural understanding isn’t far-fetched, but is already street wisdom. Why, after all, did the mathematician John W. Tukey invent the term “software” in 1957 given that the term algorithm, phonetically derived from the 9th century Persian mathematician Muhammad ibn Musa al-Khwarizmi, existed centuries before? Obviously, software referred to algorithms as control logic that was abstracted from the machine. The Unix operating system which runs on almost any kind of hardware¹ is a prime example of software as an abstraction from hardware. But just as the cultural history of computation is rich with metaphorizations and meanings inscribed into formal processes, the same is true for software. In 1970, only 13 years after Tukey’s coinage, Jack Burnham’s *Software* exhibition appropriated the term metaphorically. In a 1968 essay *Systems Esthetics*, Burnham observed that a written piece

¹The free Unix clone NetBSD supports runs almost any existing hardware platform including IBM-compatible PCs, palmtops, video game consoles, old Amiga and Atari home computers and proprietary Unix servers.

of conceptual artist Donald Judd “resembles what a computer programmer would call an entity’s /list structure/.”² Reading these semblances, Burnham did not only adopt software as a metaphor for conceptual art, but also turned, and aestheticized, computer software into concept art.

Software as practice

Just as, for example, literature is not only what is written, but all cultural practices it involves—such as oral narration and tradition, poetic performance, cultural politics—software is both material and practice. As the verb “to google” for using the Google search engine shows, or in their computational sense, “to browse,” “to chat” and “to download,” human practices are born out of the use of software. Googling is nothing but the shorthand for using the web-based client-server software written by Google corporation’s programmers. In this sense, software is no longer just machine algorithms, but something that includes the interaction, or, cultural appropriation through users. This appropriation is more than just a cybernetic human-machine interaction and what computer science and media theory often reduce to pointing, clicking and other Pavlovian responses within the restraints of a programmed system.—The same reductive understanding of interaction has turned “interactive art” in its common phenomenon of behavioral video installations into an artistic dead-end.—True interaction with technical systems involves creative use and abuse outside the box, metaphorization, writing and rewriting, configuring, disconfiguring, erasing. All these practices also make up software.

It wasn’t just artistic appropriations that inscribed metaphors into software. High-level, machine-independent programming languages and operating systems such as C and Unix gave birth, around the same time, to a culture that gradually detached software from the concept of code running on a machine. Through program code listings in books and computer magazines, source code snippets and patches exchanged in electronic networks or even oral conversations, software took up a life of its own. The results were political-philosophical movements like Free Software, programming puns such as recursive acronyms, hacker slang that mixed English and computer language constructs and poetry in computer languages such as Larry

²Jack Burnham. *Systems Esthetics*. *Artforum*, 9 1968. http://www.arts.ucsb.edu/faculty/jevbratt/classes_previous/fall_03/arts_22/Burnham_Systems_Esthetics.html. [16]

Wall's first Perl poem from 1990. Free software—in the GNU understanding of an embedded value that is not only engineering freedom, but ontological freedom—is perhaps the strongest example of a cultural and philosophical notion of software. An artistic understanding of software also abounds in computer science from Donald Knuth's *Art of Computer Programming* to Paul Graham's recent *Hackers and Painters*,³ although it might be based on a narrow understanding of art as high craftsmanship. To no longer define software as just algorithms running on hardware helps to avoid common misunderstandings of software art as some kind of of genius programmer art. If software is a broad cultural practice, then software art can be made by almost any artist.

Software versus hardware

Aside from the blurriness of software as a machine process and software as a human cultural practice, the technical distinction between software and hardware is blurry itself. Is instruction code hardware once it is burned into an EPROM, is it software when it is stored in an erasable flash ROM? What about microcode, computer programs stored right within chips in any modern CPU? Or chips like the Transmeta Crusoe which has only minimal hardware wiring and implements its CPU instruction set—like Intel-compatible x86—solely through an embedded emulation software (written originally by Linux creator Linus Torvalds)? What about the BIOS or firmware of computer mainboards, graphics cards, network adaptors without which this hardware simply isn't operational? Isn't it a totally arbitrary distinction whether a circuit is hardwired into the layout of chip transistors, or whether the same logic is stored within a memory chip? The definition of hardware, in turn, is not less doubtful. The first modern computing hardware, the Turing machine, did not materially exist, but was theoretical and imaginary. The same applies to Donald Knuth's *Mix* computer. The cultural history of computation proves that hardware can be metaphorical when algorithms run on any imagined material including the entire cosmos in Quirinus Kuhlmann's speculation. Still, in the end the distinction between software and hardware relies on Cartesian categories: Is, for example, a human brain that performs a computation a piece of hardware?

³Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, 1973-1998. [55], Paul Graham. *Hackers and Painters*. O'Reilly, Sebastopol, 2004. [40]

If the duality of software and hardware needs to be suspended, it follows that the notion of software as immaterial versus hardware as material must be suspended, too. The difference between materiality and immateriality exists within software itself: an algorithm is material in its stored, coded form and most of its cultural practices. The immaterial component might be the imagination and phantasms invested into the software, that inserting a CD-ROM, to refer to the previous example, might blow up the earth or get you a sexual turn-on (or turn-off perhaps). Some software chiefly or purely exists in imaginary form:

- vaporware, the constantly promised “stable upgrade” of a crashing computer program for example, or the ever-procrastinated new version of a piece of software, like the computer game *Duke Nukem Forever*;
- hoax viruses, i.e. E-Mail “memes” which instruct gullible readers to erase critical system files on their computer;
- imaginary virus infection; hardware failures or human mistakes wrongly interpreted as computer virus infections.

If the location or even existence of some software isn’t quite clear, if a piece software isn’t code running on machine because it appears as pseudo code in a book or is, like most Perl poetry for example, not even a working algorithm, then a technical definition of software is too limited. In the end, “software” and “computation” can’t be strictly differentiated from each other. The cultural history of software is the cultural history of computation.

Conclusion

Software, it follows, is a cultural practice made up of (a) algorithms, (b) possibly, but not necessarily in conjunction with imaginary or actual machines, (c) human interaction in a broad sense of any cultural appropriation and use, and (d) speculative imagination. Software history can thus be told as intellectual history, as opposed to media theories which consider cultural imagination a secondary product of material technology. This booklet took language computations as its primary examples mostly because language can be computational in itself. Thanks to its abstraction and grammatical structure, it also expresses computation better than any other symbolic form. Programming languages, with their modified English, are the proof. But or architecture, too, could serve as the main examples in a cultural history of computation since they both combine formal instruction with imagination.

The cultural history of computation shows that it is as rich and contradictory as that of any other symbolic form. It encompasses opposites, algorithms as a tool versus algorithms as a material of aesthetic play and speculation, computation as inner workings of nature (as in Pythagorean thought) or God (as in Kabbalah and magic) versus computation as culture and a medium of cultural reflection (starting with Oulipo and hacker cultures in the 1960s), computation as a means of abolishing semantics (Bense) versus computation as a means to structure and generate semantics (as in Lullism and Artificial Intelligence), computation as a means of generating totality (Quirinus Kuhlmann) versus computation as a means of taking things apart (Tzara, cut-ups), software as ontological freedom (GNU) versus software as ontological enslavement (Netochka Nezvanova), ecstatic computation (Kuhlmann, Kabbala, Burroughs) versus rationalist computation (from Leibniz to Turing) versus pataphysical computation as the parody of both rationalist and irrationalist computation (Oulipo and generative psychogeography), algorithm as expansion (Lullism, generative art) versus algorithm as constraint (Oulipo, net.art), code as chaotic imagination (Jodi, codeworks) versus code as structured description of chaos (Tzara, John Cage).

Computation and its imaginary are rich with contradictions, and loaded with metaphysical and ontological speculation. Underneath those contradictions and speculations lies an obsession with code that executes, the phantasm that words become flesh. It remains a phantasm, because again and again, the execution fails to match the boundless speculative expectations invested into it. Cultural and political semantics result merely from its dull formalisms and their interference with daily life, from account balance statements to “end-user software.” Formalisms create semantics in a wholly different way than people expect from an allegedly “intelligent machine.” Computers therefore exist, as hacker wisdom says, to solve problems which we would not even be aware of having if not for the computers themselves.

References

- [1] Theodor W. Adorno and Max Horkheimer. *Dialectic of Enlightenment*. Verso, London, 1979 (1947). 83
- [2] Andreas Alciatus. *Emblematum Libellus*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1991 (1542). 22
- [3] Mario Alinei and Alfredo Schiaffini. *Spogli elettronici dell'italiano delle origini e del Duecento*. Mouton, The Hague, 1968. 73
- [4] Johann Heinrich Alsted. *Encyclopaedia*. Holzboog, Stuttgart (Herborn), 1989 (1630). 45
- [5] Johann Valentin Andreae. *Turris Babel*. Zetzner, Strasbourg, 1619. 51
- [6] Johann Valentin Andreae. *Fama Fraternitatis, Confessio Fraternitatis, Chymische Hochzeit: Christiani Rosencreutz Anno 1459*. Calwer Verlag, Stuttgart, 1994 (1973). 51
- [7] Jonathan Barnes, editor. *Early Greek Philosophy*. Penguin, Harmondsworth, 2002. 20
- [8] Walter Benjamin. *Charles Baudelaire: A Lyric Poet in the Era of High Capitalism*. New Left Books, London, 1973. 71
- [9] Mercedes Blanco. *Les rhétoriques de la pointe*. Librairie Honoré Champion, Paris, 1992. 22, 24
- [10] Harold Bloom, editor. *Selected Writings of Walter Pater*. Columbia University Press, New York, 1982. 22
- [11] Boethius. De Musica. In *Opera omnia*. Firmin-Didot, Paris, 1882. 21
- [12] Anthony Bonner, editor. *Doctor Illuminatus: A Ramon Llull Reader*. Princeton University Press, Princeton, New Jersey, 1993. 24, 133
- [13] Jorge Luis Borges. The Library of Babel. In *Ficciones*, pages 79–88. Grove Press, New York, 1941. 61
- [14] André Breton. Manifesto of Surrealism. In *Manifestoes of Surrealism*, pages 1–48. Ann Arbor Paperbacks, Ann Arbor, Michigan, 1924. 71
- [15] Edmund Burke. *A Philosophical Enquiry into the Origin of our Ideas of the Sublime and Beautiful*. Oxford University Press, Oxford, 1990 (1757). 14
- [16] Jack Burnham. Systems Esthetics. *Artforum*, 9 1968. http://www.arts.ucsb.edu/faculty/jevbratt/classes_previous/fall_03/arts_22/Burnham_Systems_Esthetics.html. 122
- [17] William S. Burroughs. The Cut-Up Method of Brion Gysin. In *The Third Mind* [18]. 77, 80
- [18] William S. Burroughs. *The Third Mind*. Viking, New York, 1978. 127, 128
- [19] William S. Burroughs. *Electronic Revolution*. Expanded Media Edition, Bonn, 1982. 18, 20, 50
- [20] John Cage. *Roaratorio. Ein irischer Circus über Finnegans Wake*. Athenäum, Königstein/Taunus, 1982. 77, 78

- [21] Italo Calvino. *Invisible Cities*. Harvest Books, Fort Washington, 1978. 81
- [22] Italo Calvino. Cybernetics and Ghosts. In *The Uses of Literature*, pages 3–27. Harcourt, San Diego, 1982 (1967). 80, 92
- [23] Italo Calvino. Prose and anticombinatorics. In Motte [71], pages 143–152. 92
- [24] Italo Calvino. *If on a Winter's Night a Traveller*. Everyman Publishers, London, 1993 (1979). 73
- [25] Vuk Cosic. *Contemporary ASCII*. Kapelica, Ljubljana, 2000. 112
- [26] Aleister Crowley. *The Book of Lies*. Red Wheel Weiser, 1970 (1913). 48
- [27] Franz Josef Czernin and Ferdinand Schmatz. Notes about the Poetry Program POE, 1990. http://www.aec.at/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=8950. 112
- [28] Richard Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, 1989 (1976). 19
- [29] Franz Dornseiff. *Das Alphabet in Mystik und Magie*. Teubner, Leipzig, Berlin, 1925. 15
- [30] Umberto Eco. *Der Streit der Interpretationen*. Universitätsverlag Konstanz, Konstanz, 1987. 25
- [31] Umberto Eco. *Foucault's Pendulum*. Ballantine Books, 1990. 33
- [32] Umberto Eco. *Misreadings*. Harvest, San Diego, 1993. 82
- [33] Umberto Eco. *The Island of the Day Before*. Penguin, Harmondsworth, 1996. 23
- [34] Hans Magnus Enzensberger. *Einladung zu einem Poesie-Automaten*. Suhrkamp, Frankfurt/M., 2000. 111
- [35] James George Frazer. *The Golden Bough*. Macmillan, London, 1950. 19
- [36] Franchino Gaffurio. *De harmonia musicorum*. Forni, Bologna, 1972 (1518). 21, 133
- [37] Joscelyn Godwin. *Athanasius Kircher*. Edition Weber, Berlin, 1994 (1979). 35, 133
- [38] Eugen Gomringer. 3 variationen zu kein fehler im system. In Eugen Gomringer, editor, *konkrete poesie*, pages 63–64. Reclam, Stuttgart, 1972. 65
- [39] Eugen Gomringer. vom vers zur konstellation. In Eugen Gomringer, editor, *zur sache der konkreten*, volume 1, pages 7–12. Erker-Verlag, St. Gallen, 1988 (1954). 66
- [40] Paul Graham. *Hackers and Painters*. O'Reilly, Sebastopol, 2004. 123
- [41] Mark Greengrass, editor. *The Hartlib Papers*. University Microfilms, Michigan, 1996. CD-ROM. 60
- [42] Brion Gysin. Permutation poems. In *The Third Mind* [18]. 17
- [43] Georg Philipp Harsdörffer. *Frauenzimmer Gesprächspiele*. Deutsche Neudrucke: Reihe Barock. Niemeyer, Tübingen, 1968-69 (1643-57). 58
- [44] Georg Philipp Harsdörffer. *Mathematische und philosophische Erquickstunden*. Texte der frühen Neuzeit. Keip, Frankfurt (Nürnberg), 1990 (1636). 58, 133
- [45] Charles O. Hartman and Hugh Kenner. *Sentences*. Sun and Moon Pres, Los Angeles, 1995. 75
- [46] Douglas R. Hofstadter. *Gödel, Escher, Bach*. Basic Books, New York, 1979. 26
- [47] Sharon Hopkins. Camels and Needles : Computer Poetry meets the Perl Programming Language, 1991. <http://www.wall.org/~sharon/plpaper.ps>. 95

- [48] Moshe Idel. *The Mystical Experience in Abraham Abulafia*. State University of New York Press, Albany, 1988. 48
- [49] Moshe Idel. Ramon Lull and Ecstatic Kabbalah. *Journal of the Warburg and Courtauld Institutes*, 51:170–174, 1988. 36
- [50] Roman Jakobson. Two Aspects of Language and Two Types of Aphasic Disturbances. In *Fundamentals of Language*, pages 115–133. Mouton, The Hague, Paris, 1971. 19
- [51] Alfred Jarry. *Exploits and Opinions of Dr. Faustroll, Pataphysician*. Exact Change, Berkeley, 1996 (1911). 88
- [52] Asger Jorn. La création ouverte et ses ennemis. In *Situationniste [93]*, pages 175–196. 71
- [53] Immanuel Kant. *Critique of the Power of Judgment*. The Cambridge Edition of the Works of Immanuel Kant in Translation. Cambridge University Press, Cambridge, 2001. 14
- [54] Alan Kay. User Interface: A Personal View. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 191–207. Addison Wesley, Reading, Massachusetts, 1990. 113
- [55] Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, 1973-1998. 27, 123
- [56] Donald E. Knuth. *Things a Computer Scientist Rarely Talks About*. CSLI Publications, Stanford, 2001. 27
- [57] Jan Amos Komenský. Orbis sensualium pictus. In *Opera Omnia*, volume 17, pages 69–271. Academia Praha, Praha, 1970. 40, 133
- [58] Quirinus Kuhlmann. *Epistolae duae*. Lotho de Haes, Amsterdam, 1674. 105
- [59] Quirinus Kuhlmann. *Prodomus*. Lotho de Haes, Amsterdam, 1674. 61
- [60] Quirinus Kuhlmann. *Himmlische Libes-küsse*. Niemeyer, Tübingen, 1971 (1671). 47
- [61] François Le Lionnais. Lipo: First manifesto. In *Motte [71]*, pages 26–28. 74, 94
- [62] G.W. Leibniz. Dissertatio de arte combinatoria. In *Sämtliche Schriften*, volume 1 of VI, pages 165–230. Akademie-Verlag, Berlin, 1989. 45
- [63] Steven Levy. *Hackers*. Project Gutenberg, Champaign, IL, 1986 (1984). 27, 94
- [64] Klaus Maeck and Walter Hartmann, editors. *Decoder Handbuch*. Trikont, Duisburg, 1984. 18
- [65] Solomon Maimon. *An Autobiography*. University of Illinois Press, Chicago, 2001 (1792). 29
- [66] Stéphane Mallarmé. *Un coup de dés jamais n'abolira le hasard*. Gallimard, Paris, 1993 (1914). 64
- [67] Katharine Mieszkowski. The most feared woman on the internet. *Salon.com*, 2002. <http://www.salon.com/tech/feature/2002/03/01/netochka/>. 114
- [68] Abraham A. Moles. erstes manifest der permutationellen kunst. Stuttgart, 1963. 92
- [69] Abraham A. Moles. *Art et Ordinateur*. Casterman, Paris, 1981 (1971). 92
- [70] Daniel Georg Morhof. *De Acuta Dictione*. Petrus Böckmannus, Lübeck, 1705. 58
- [71] Warren F. Motte, editor. *Oulipo. A Primer of Potential Literature*. University of Nebraska Press, Lincoln, London, 1986. 128, 129, 130

- [72] Novalis. *Das Allgemeine Brouillon*. Meiner, Hamburg, 1993 (1798/99). 63
- [73] Hans Otto Peitgen. *The Beauty of Fractals*. Springer, Heidelberg, New York, 1986. 28
- [74] Georges Perec. *Die Maschine*. Reclam, Stuttgart, 1972 1968. 134
- [75] Georges Perec. *A Void*. HarperCollins, New York, 1994 1969. 91
- [76] Giovanni Pico della Mirandola. *Oration on the Dignity of Man*. MacMillan, 1985 (1486). 35
- [77] Publilius Optatianus Porfyrius. *Publili Optatiani Porfyrii Carmina*. Paravia, Turin, 1973. 44
- [78] Vladimir Propp, editor. *Morphology of the Folktale*. University of Texas Press, Austin, Texas, 1968 (1927). 81
- [79] Thomas Pynchon. *Gravity's Rainbow*. Vintage, London, 1995 (1973). 33
- [80] Thomas Pynchon. *The Crying of Lot 49*. Perennial Classics, New York, 1999 (1967). 33
- [81] Raymond Queneau. *Exercices de style*. Gallimard, Paris, 1947. 89
- [82] Raymond Queneau. *Cent mille milliards de poèmes*. Gallimard, Paris, 1961. 89, 90
- [83] Raymond Queneau. Potential literature. In Motte [71], pages 51–64. 90
- [84] Simon Richter. *Laocoon's Body and the Aesthetics of Pain: Winckelmann, Lessing, Herder, Moritz, Goethe*. Wayne State University Press, Detroit, 1992. 22
- [85] W. Rhys Roberts, editor. *Longinus on the Sublime*. Cambridge University Press, Cambridge, 1899. 14
- [86] Friedrich Rückert. *Grammatik, Poetik und Rhetorik der Perser*. Verlagsbuchhandlung Otto Zelle, Antiquariat Otto Harrassowitz, Wiesbaden, Osnabrück (Gotha), 1966 (1874). 41
- [87] Maciej Kazimierz Sarbiewski. De Acuto et Arguto liber unicus. In *Wykladi Poetyki*, pages 1–20. Wydawnictwo Polskiej Akademii Nauk, Wrocław, Krakow, 1958. 23, 133
- [88] Tom Sawyer and Arthur David Weingarten. *Plots Unlimited*. Ashleywilde, Malibu, 1994. 82, 85
- [89] Julius Caesar Scaliger. *Poetices libri septem*. Frommann, Stuttgart, 1964 (1561). 44
- [90] Jacques Scherer. *Le livre de Mallarmé*. Gallimard, Paris, 1977 (1957). 64
- [91] John R. Searle. Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3:417–456, 1980. 107, 134
- [92] Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, July 1948. 68, 74
- [93] Internationale Situationniste, editor. *Internationale situationniste. Édition augmentée*. Librairie Arthème Fayard, Paris, 1997 (1958-1969). 89, 93, 129
- [94] Cornelia Sollfrank, editor. *net.art generator*. Verlag für moderne Kunst, Nürnberg, 2004. 87, 134
- [95] Alan Sondheim. Introduction: Codework. *American Book Review*, 22(6):1–4, September 2001. 98
- [96] Karlheinz Stockhausen. Weberns Konzert für neun Instrumente op. 24. In *Texte zur elektronischen und instrumentalen Musik*, pages 24–31. M. DuMont Schauberg, Köln, 1963 (1953). 26
- [97] Jonathan Swift. *Gulliver's Travels*. Washington Square Press, New York, 1960. 60, 134

- [98] Tristan Tzara. Pour fair une poème dadaïste. In *Oeuvres complètes*. Gallimard, Paris, 1975. 76
- [99] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl*. O'Reilly, Cambridge, Köln, Paris, Sebastopol, Tokyo, 1996. 75
- [100] McKenzie Wark. Essay: Codework. *American Book Review*, 22(6):1–5, September 2001. 97
- [101] Frances Yates. *The Art of Memory*. Routledge & Kegan Paul, London, 1965. 43
- [102] La Monte Young. Composition 1960 #10 to Bob Morris. In Harald Szeemann and Hans Sohm, editors, *happening & fluxus*. Kölnischer Kunstverein, Köln, 1970 (1960). 49

List of Figures

1	Eric Bogosian as mad scientist Travis Dane in the film <i>Under Siege 2—Dark Territory</i> , Warner Brothers/Regency Enterprises, 1995	7
1	Google search results for the keyword combination “software” and “magic”	16
2	Franchino Gaffurio, <i>De Harmonia Musicorum</i> , Milan, 1518 [36], title page illustration	21
3	Maciej Kazimierz Sarbiewski, <i>De acuto et arguto</i> , 1628 [87], diagram illustrating the construction of a witty statement	23
4	<i>Torah Codes 2000</i> , a commercial software application for Kabbalistic text analysis, screenshot from the online store of http://www.jewishsoftware.com	33
5	Athanasius Kircher, <i>Oedipus Aegyptiacus</i> , 1652-4, sunflower-shaped diagram of the name of God in different languages [37]	34
6	Ramon Llull (Raimundus Lullus), <i>Ars brevis</i> , 1308 [12], the four <i>figurae</i> , or algorithms, for processing the table (<i>tabula</i>) of elements	36
7	Jan Amos Komenský (Comenius), <i>Orbis pictus</i> , 1657 [57]. Correlating sounds, images and letters on its opening pages, the book creates a graphical user interface for written language	40
8	The three-dimensional <i>Looking Glass</i> desktop, an experimental software development project of Sun Microsystems	42
9	Georg Philipp Harsdörffer, <i>Fünffacher Denckring der teutschen Sprache</i> , 1636 [44], a morphological generator of German words	46

1	<i>xlife</i> , a free software implementation of <i>Conway's Game of Life</i> for Unix-like operating systems	57
2	Jonathan Swift, <i>Gulliver's Travels</i> , 1726 [97], original illustration of the combinatorial "engine" at the Grand Academy of Lagado	59
3	Cornelia Sollfrank et.al., <i>Net.art generators</i> [94], automatically generated variations of Andy Warhol's <i>Flowers</i>	84
4	Adrian Ward, <i>Auto-Illustrator</i> , screenshot of the random text tool	86
1	Athanasius Kircher, <i>Musurgia universalis</i> , 1650, design of an algorithmic music composition machine	105
2	John Searle, the Chinese room thought experiment [91] as illustrated on the website http://www.macrovu.com/CCTMap4ChineseRm.html	107
3	Georges Perec, <i>Die Maschine</i> , 1968 [74], script pages of the radio play	110

Index

- .walk, 70
- 3dwm (software), 41

- Abulafia, Abraham, 3, 30, 31, 46, 47, 50
- Acid (software), 83
- Adobe (company), 83
- Adobe Illustrator (software), 83
- Adorno, Theodor W., 81
- aesthetics, 18, 20, 26, 65, 66, 68, 71, 74, 77, 81, 83, 86, 91, 96, 110, 111, 114
- al-Khwarizmi, Muhammad ibn Musa, 119
- ALAMO, 89
- Albers, Josef, 68
- Alciatus, Andreas, 20
- aleatory, 88, 101
- Algol (programming language), 92
- algorithm, 6, 7, 15, 21, 23–25, 29, 30, 36, 37, 42, 44, 46, 48, 51–53, 55, 56, 63, 64, 67, 71–84, 88–91, 101–104, 106–111, 113, 119–123
- Alinei, Mario, 71
- Alsted, Johann Heinrich, 37, 38, 43, 61
- Amiga (operating system), 119
- anarchism, 75, 77, 116
- Andrae, Johann Valentin, 38, 49, 50, 58, 59
- Anger, Kenneth, 14
- Antonioni, Michelangelo, 80
- Apple Macintosh (operating system), 39, 41, 73, 85
- Apple Newton, 5
- architecture, 10, 63, 65, 122

- Aristophanes, 10
- Aristotle; Aristotelian, 22, 27, 57
- Arnaud, Noël, 92
- Art and Language, 67
- Artificial Intelligence (A.I.), 36, 53, 70, 72, 81, 85, 86, 91, 102–106, 110, 123
- artificial life, 85
- ASCII art, 96, 110
- assembly (programming language), 36
- AT&T (company), 65
- Atari (company), 119
- Athenaios, 40, 43
- Auto-Illustrator (software), 83, 84
- avant-garde, 24, 65, 66, 87

- Böhme, Jakob, 46
- Bach, Johann Sebastian, 24, 48, 73
- Band-in-a-Box (software), 83
- Barthes, Roland, 78
- BASIC (programming language), 31
- Baudelaire, Charles, 69
- Bauhaus, 63, 65, 68, 83
- Beatles, 14
- Bell Labs, 65
- Benjamin, Walter, 69
- Bense, Max, 65–72, 85, 88–91, 109, 110, 123
- Bible, 13, 32, 45
- Bill, Max, 63, 68
- BIOS, 121
- Blade Runner (film), 106
- bloatware, 7
- Boethius, Anicius Manlius Severinus, 18, 19
- Bond, James, 6

- Borges, Jorge Luis, 59–62, 64, 73, 102
- Boulez, Pierre, 24, 77
- Boyle, Robert, 58
- Bremer, Claus, 64
- Breton, André, 69, 87
- Brown, Earle, 75
- Bruno, Giordano, 37, 52
- Burke, Edmund, 12
- Burnham, Jack, 67, 119, 120
- Burroughs, William S., 15–18, 26, 46, 47, 50, 75, 77, 88, 89, 115, 123
- BYTE (computer magazine), 72
- C (programming language), 75, 120
- Cage, John, 7, 75–78, 84, 88, 101, 115, 123
- Calvino, Italo, 71, 78, 79, 85, 90, 91, 109
- Campanella, Tommaso, 38, 50, 58
- Cardew, Cornelius, 25
- Carroll, Lewis, 11, 12
- Cathars, 6
- Caulfield, Patricia, 81
- cellular automata, 55
- chaos, 10, 26, 30, 74, 77, 79, 123
- chat roboter, 70
- Chomsky, Noam, 101
- Cobra (group), 92
- codework, 97
- codeworks, 11, 96–99, 101, 102, 108, 112, 113, 116, 117, 119, 123
- Collège de Pataphysique, 87, 89, 92
- combination, 29, 30, 36, 37, 44, 51, 52, 79, 85, 87, 115
- combinatorics, 43, 46, 57, 60–62, 78–80, 89–91, 101–103
- Comenius, Jan Amos, 38, 39, 41, 49, 50, 59
- concrete poetry, 63–66, 68, 101, 109
- Conrad, Tony, 16
- constellation, 62–65
- constraint, 43, 47, 88–92, 96, 111, 123
- Conway's Game of Life, 55, 56
- Conway, John Horton, 56
- Cosic, Vuc, 110
- CPU, 48, 121
- Cronenberg, David, 47, 116
- Crowley, Aleister, 14–16, 46, 113, 114
- cryptography, 6, 7
- Culver, Andrew, 75, 76
- cut-ups, 10, 15–17, 68, 75, 77, 78, 88, 101, 115, 123
- cybernetics, 67, 68, 90, 91, 111, 114, 120
- cyberpunk, 12
- Czernin, Franz Josef, 73, 110
- d'Alembert, Jean-le-Rond, 37
- Döhl, Reinhard, 109
- Dada, 74–77, 81, 82, 84, 101
- dadadodo (software), 73
- database, 13, 32, 35, 36, 80
- Dawkins, Richard, 17
- de Jong, Jacqueline, 92
- Debord, Guy, 91, 92
- Deconstructor (software), 73
- demiurgy, 52, 55
- dice, 30, 55, 62, 76, 85
- Dick, Philip K., 106
- Dicuil, 42
- Diderot, Denis, 37
- Dissociated Press (software), 73
- Dornseiff, Franz, 13
- DOS (operating system), 39, 73, 75
- Duke Nukem Forever (software), 122
- E-Mail, 10, 95, 97, 108, 122
- Eco, Umberto, 21–23, 31, 79, 80
- ecstatic, 16, 26, 30, 34, 46, 51, 75, 77, 123
- Einstein, Albert, 30
- Emacs (software), 48, 73
- emblems, 20
- encyclopedism, 37, 38, 51, 61, 79, 84, 87
- Enzensberger, Hans Magnus, 109, 110
- Essl, Karlheinz, 73
- Feldman, Morton, 75, 115
- Fenollosa, Ernest, 66
- Fluxus, 25, 47, 67, 73, 75
- Flynt, Henry, 25
- Fourier, Jean Baptiste Joseph, 23
- fractals, 26

- Frazer, James George, 16–18
 Free Software, 48–50, 52, 120
 Freemasons, 6
 Frelih, Luka, 81

 Gödel, Kurt, 60, 64
 Gaffurio, Franchino, 19, 21
 Gardner, Martin, 11
 genius, 22, 42, 57, 74, 80, 82, 104, 121
 George, Stefan, 66
 gnosticism, 13, 46, 114
 GNU, 6, 13, 48, 49, 73, 103, 121, 123
 GNU General Public License (GPL), 48, 49
 God, 12, 13, 16, 25, 27, 28, 30, 31, 33–37, 39, 42, 50, 123
 Godard, Jean-Luc, 80
 Goethe, Johann Wolfgang, 107–109, 111
 golem, 39
 Gomringer, Eugen, 63, 64, 66
 Google, 13, 120
 gothic, 12
 Gracián, Baltasar, 20, 22
 Graham, Paul, 121
 Grand Guignol, 79
 Gysin, Brion, 15–17, 26, 29, 46, 75, 77, 88, 89, 92

 Haacke, Hans, 67
 hacker, 25, 48, 92, 96, 109–111, 120, 123
 hardware, 10, 52, 67, 70, 103, 106, 119, 121, 122
 harmony, 18–21, 23, 25, 26
 Harsdörffer, Georg Philipp, 43–45, 52, 53, 56, 59, 79, 80, 88, 109
 Hartlib, Samuel, 49, 58
 Hartman, Charles O., 73
 Hawking, Stephen, 30
 Heemskerck, Joan, 93
 Heidegger, Martin, 85
 Heraclitus, 18, 19
 hermetic, 23, 38, 49
 hieroglyphs, 6, 7, 33
 Hofstadter, Douglas R., 24, 48
 Honeywell (company), 46
 Hopkins, Sharon, 93

 Horkheimer, Max, 81
 Hubbard, L. Ron, 113–115

 I Ching, 75, 76
 I/O/D, 86, 89, 115, 116
 IBM (company), 119
 IC (software), 75
 Idel, Moshe, 34
 indeterminism, 75–78, 112
 industrial music, 14, 114, 115
 information aesthetics, 65, 71, 91, 109
 Intel (company), 121
 interactive art, 93, 112, 120
 interface, 24, 36, 38, 41, 50, 52, 83, 85, 93, 110–112, 115, 117
 Invisible College, 58
 IRCAM, 24
 Isou, Isidore, 89
 Itten, Johannes, 68

 Jagger, Mick, 28
 Jakobson, Roman, 12, 17
 Jarry, Alfred, 70, 86, 88
 Jodi, 86, 93–98, 110–112, 116, 123
 John, Elton, 28
 Johnston, Ryan, 81
 Jorn, Asger, 68, 69, 87
 Joy, Bill, 6
 Joyce, James, 10–12, 72, 76
 Judd, Donald, 120

 Kabbalah, 27–31, 33–35, 39, 44–46, 48, 50, 62, 69, 84, 101, 114, 123
 Kafka, Franz, 72
 Kant, Immanuel, 12, 27, 116
 Kawara, On, 67
 Kay, Alan, 111
 Kenner, Hugh, 72, 73
 Khlebnikov, Velimir, 12, 66
 Kircher, Athanasius, 33, 34, 37, 45, 103, 104, 109
 Knuth, Donald, 6, 25, 121
 Korzybski, Alfred, 114
 Kosuth, Joseph, 67
 Kristeva, Julia, 78
 Kuhlmann, Quirinus, 45–47, 50–53, 56, 59–62, 103, 104, 106, 121, 123

- Kyburz, Hanspeter, 7
- Lévi-Strauss, Claude, 78, 80
- Lansius, Thomas, 43
- Le Lionnais, François, 87, 88, 92
- leet speech, 96, 113
- Leibniz, Gottfried Wilhelm, 37, 39, 43, 46, 61, 66, 91, 123
- Leiris, Michel, 87
- Lemaître, Maurice, 89
- Leopold, Richard, 81, 82
- Lessing, Gotthold Ephraim, 20
- Lettrism, 69, 89
- Levy, Steven, 25
- linguistics, 17, 27, 35, 43–45, 56, 67, 71, 78, 79, 87, 101, 102, 114, 115
- Lippard, Lucy, 67
- Lisp (programming language), 48, 102, 103
- Llull, Ramon, 22, 34–37, 39, 43, 44, 46, 51–53, 69, 73, 80, 84, 103, 109
- Logo (programming language), 56
- Looking Glass (software), 41
- Love, Courtney, 28
- Lovecraft, H.P., 7
- Lullism, 37–39, 43, 45, 46, 51, 52, 56–59, 61, 62, 66, 71, 79, 84, 87, 88, 91, 101–104, 106, 123
- Lutz, Theo, 72, 92
- MacLow, Jackson, 73
- macrocosm, 23, 43, 45, 52, 55, 62, 66, 74, 75, 86, 104
- Madonna, 28
- magic, 6, 7, 12–18, 26–29, 33, 39, 44, 52, 70, 75, 77, 84, 85, 89, 101, 114, 115, 119, 123
- Magix Musicmaker (software), 13
- Maimon, Solomon, 27, 28
- Mallarmé, Stéphane, 62–64, 66, 74, 76, 86
- malware, 7
- Mark V. Chaney (software), 73
- Markov chains, 72, 73, 82, 101, 110
- Markov, Andrei, 72
- MAX (software), 23, 24, 113
- McKenzie, Andrew, 114, 115
- McLuhan, Marshall, 11, 85, 111
- media theory, 11, 85, 86, 91, 95, 120, 122
- meme, 17, 122
- Mersenne, Marin, 104
- Mesolist (software), 76
- metaphysics, 7, 39, 46, 49, 52, 62, 63, 66, 67, 74, 75, 86, 89, 106, 123
- mez, 9, 10, 12, 97, 98, 116, 117
- mezangelle, 9, 10
- microcosm, 23, 43, 52, 55, 62, 66, 74, 75, 86, 104
- Microsoft (company), 114
- Microsoft Windows (operating system), 41, 93, 114
- MIT (Massachusetts Institute of Technology), 25, 92, 103
- Mix computer, 121
- Moles, Abraham M., 67, 69, 90, 91
- Morhof, Daniel Georg, 56, 104
- Mozilla (software), 73
- music, 7, 18, 19, 23–25, 62, 75–77, 83, 88, 91, 101, 104, 110, 112, 113, 122
- mysticism, 13, 25, 27, 33, 34, 39, 46, 91
- Napoleon, 6, 33
- Native Instruments (company), 83
- NATO (software), 113
- Neoplatonism, 13, 33, 35, 62
- net.art, 9, 11, 81, 82, 93, 96, 97, 110–112, 116, 117, 123
- Newton, Isaac, 57
- Nezvanova, Netochka, 112–114, 116, 123
- Novalis, 61, 62, 66, 86, 91
- O'Rourke, Joseph, 72
- occult, 13–16, 18, 25, 26, 28, 31, 33, 45, 46, 52, 61, 66, 84, 113–116
- Olmi, Ermanno, 80
- ontology, 26, 49, 55, 73, 76–78, 80, 85, 106, 112, 121, 123
- Optatianus Porfyrius, 42, 43, 74
- Oulipo, 70, 72, 87–93, 96, 107, 109–111, 123
- P-Orridge, Genesis, 14, 16

- Paesmans, Dirk, 93
 Paik, Nam June, 11, 111
 Partition Magic (software), 13
 pataphysics, 70, 86–88, 107, 110, 111, 123
 Pater, Walter, 20
 Peirce, Charles S., 65
 Peitgen, Hans Otto, 26
 Perec, Georges, 89, 90, 107–111
 Perl (programming language), 73, 92, 93, 121
 Perl poetry, 73, 122
 permutation, 15, 17, 24, 29–31, 33, 34, 37, 39, 40, 42, 43, 45, 46, 48, 51, 52, 62–64, 73, 74, 77, 78, 82, 90, 91, 103, 110, 114
 Photoshop (software), 83
 Pico della Mirandola, Giovanni, 33, 34
 Plato, 40, 59, 85
 Plots Unlimited (software), 80, 83
 Plunderphonics, 76
 POE (software), 73, 110
 poetics, 10, 12, 15, 17, 20, 39, 42, 43, 45, 47, 51, 56, 64, 66, 68, 74, 75, 78, 80, 82, 84–86, 88–92, 102, 107, 108
 Pound, Ezra, 66
 Prehn, Ralf, 81
 Propp, Vladimir, 78–80
 Pseudo-Longinus, 12
 psychogeography, 69, 70, 76, 79, 86, 90, 123
 Puckette, Miller, 24
 Pure Data (software), 24
 Pynchon, Thomas, 30
 Pythagorean, 18–27, 33, 39, 62, 66, 69, 74–76, 84, 123
- Queneau, Raymond, 87, 88, 90, 109
- random, 7, 22, 62, 72–76, 78, 83
 Raskin, Jef, 85
 recursion, 48, 49, 51, 62, 64, 73, 77, 78, 98, 102
 recursive acronym, 48, 120
 religion, 7, 13, 28, 35, 39, 55, 61, 77, 87
 reverse-engineering, 30, 39, 45, 51, 59
 rhetoric, 12, 20–22, 35, 39–45, 51, 52, 56, 104
 Rolling Stones, 14
 romanticism, 12, 22, 26, 61, 66, 69, 70, 74, 76, 79, 80, 87, 89
 Rosenberg, Jim, 76
 Rosetta stone, 6, 33
 Rosicrucian, 45, 49, 50, 113
- Sarbiewski, Kazimierz, 20, 21, 104
 satanism, 46
 Saussure, Ferdinand de, 27
 Scaliger, Julius Caesar, 42, 43, 64, 74
 Schönberg, Arnold, 24
 Schiller, Friedrich, 12
 Schmatz, Ferdinand, 73, 110
 Scholem, Gershom, 28
 Schwitters, Kurt, 73
 science fiction, 6, 7, 12, 106, 116
 Scientology, 113–115
 scratching, 76
 Seagal, Steven, 5, 6, 10, 119
 Searle, John R., 104–106, 108
 Sefer Yetzirah, 28–31, 34, 36, 37, 39, 45, 52, 62, 74, 119
 Sefirot, 28, 33, 34
 semantics, 10, 16, 18, 21, 35–37, 41, 42, 46, 48, 51–53, 61, 65, 66, 68, 69, 72, 77, 84, 86, 89, 91, 98, 102, 104, 107, 108, 110, 113, 114, 123
 semiotics, 65–67, 78, 80
 serial music, 24, 29, 77
 serial music, 24
 Shakur, Tupac, 96
 Shanken, Edward A., 67
 Shannon, Claude, 65, 66, 71–73
 Shulgin, Alexei, 113
 Situationism, 68–70, 79, 86, 87, 89, 91, 92
 Snow, C. P., 14
 socialfiction.org, 70
 Sollfrank, Cornelia, 81, 82, 85, 86
 Solomon, 45, 51
 Sommerville, Ian, 15
 Sondheim, Alan, 96–98
 Sonic Foundry (company), 83

- Spears, Britney, 28, 77
speculative, 6, 7, 12, 17, 27, 30, 33,
34, 58–61, 70, 71, 90, 91, 107,
115, 119, 121–123
Stallman, Richard, 6, 48
stochastics, 30, 72, 76–78, 109, 112
Stockhausen, Karlheinz, 24, 25, 77,
84, 88
structuralism, 17, 67, 78–80, 90
Sun (company), 41
Surrealism, 68–70, 75, 79, 86, 87, 93
Suzuki, Daisetz T., 75
Swift, Jonathan, 57–59, 61, 71, 73,
74, 86, 89, 102, 106
syntax, 9, 21, 41, 42, 52, 53, 91, 92,
98, 102, 105, 108, 113
- Taylor, Liz, 28
Tel Quel, 78, 90
Terminator (movie), 6
Tesauro, Emanuele, 20, 22
TeX (software), 6, 25
TextMangler (software), 73
Theall, Donald, 11
theosophy, 33, 37, 46, 55, 86, 89, 104
theurgy, 13, 27, 28, 39, 43, 44, 52
Thoens, Barbara, 81
Tillotson, John, 13
Torah, 27, 30–32, 101
Torvalds, Linus, 121
Traktor (software), 83
Transmeta (company), 121
travesty (software), 72, 73
Tukey, John W., 119
Turing, Alan, 30, 56, 70, 88, 106, 121,
123
Tzara, Tristan, 74–78, 81, 82, 101,
123
- Ullrichs, Tim, 64
Unix (operating system), 9, 13, 39,
41, 48, 92, 96–98, 110, 119, 120
- vaporware, 7, 122
vi (software), 6
Vian, Boris, 87
Vienet, René, 79
virtual reality, 85, 86, 111, 116
virus, 7, 17, 18, 47, 49, 94, 95, 108,
122
Visconti, Luchino, 80
Voynich Manuscript, 6, 7, 10
- Wagner, Richard, 62
Wall, Larry, 72, 92, 120
Ward, Adrian, 83, 86
Warhol, Andy, 81
Wark, McKenzie, 95
Web Stalker (software), 115, 116
Weiner, Lawrence, 67
Wiki (software), 70
Wilkins, John, 58
Wilson, Colin, 7
Winckelmann, Johann Joachim, 20
Wolff, Christian, 75
- X11 Window System (software), 41
Xerox PARC, 20, 85, 111
xlife (software), 55
- Young, La Monte, 47
- Zawinsky, Jamie, 73
Zen, 75
ZKM (media arts center), 68