2

Bend Sinister: Monstrosity and Normative Effect in Computational Practice

Simon Yuill

To learn programming is to embark on years of practice, learning to engage with the unknowable, while battling with complex and sometimes unhelpful theory.

Alex McLean, 'Exclusion in Free Software Culture'[1]

What pleasure does programming give? This daunting summary given by Alex McLean neatly summarizes all that may dissuade someone from taking up the challenge, although for others it may be all the incentive required. Beyond mere masochism, or intellectual bravado, in what forms does pleasure arise in the labour of coding? A common response is that offered under the guise of *elegance*, that there is a certain satisfaction derived from writing an algorithm that performs its task in a particularly effective and clear manner. Such elegance is often postulated as something understood *aesthetically*, as something that is both an intuitive sensuous response and also a matter of cultivated taste. Such is the position put forward in Donald Knuth's *The Art of Computer Programming*. For McLean this pleasure is found by coding in nightclubs. *feedback.pl* is a Perl script by McLean that enables a programmer to perform live improvised music by writing and editing simple algorithms that are executed in realtime – a practice known as *livecoding*. In providing McLean a means of colliding programming and dancing, the xterm[2] and techno, *feedback.pl* unleashes a previously unknown pleasure that situates the arduous labour of coding within the hedonistic monstrations of clubbing: 'with hundreds of people dancing to my Perl, jumping about to my subroutines, whooping as I started up a new script'.[3] As such it challenges many norms of programming practice and societal expectations of what programming is, and reverses the established (non) physicality of the programmer.[4] Even while making this challenge, *feedback.pl* remains a highly skilful exemplification of Knuth's elegant coding which, as Knuth describes it, 'can be an aesthetic experience much like composing poetry or music'.[5] McLean's program itself is brief and conceptually succinct. In an article presenting the project, *Hacking Perl in Nightclubs*, McLean provides a coding example based around a modulo operation on a divisor of four, creating a basic four-four rhythm against which more complex polyrhythms are played, both encapsulating and transcending the quintessence of modern dance music in a few short lines of code:[6]

```
sub bang {
    my $self = shift;
    my $note = 100;
    $note += 50 if $self->{bangs} % 4 == 0;
    $note -= 30 if $self->{bangs} % 3 == 0;
    $note += 60 if $self->{bangs} % 7 == 0;
    beep($note, 40);
    $self->code->[0] = '# note: ' . $note;
    $self->modified;
}
```

But things could turn another way. With minor modifications a program such as *feedback.pl* could direct the programmer's text not to the Perl interpreter but in some other direction such as /dev/dsp, the basic audio output on UNIX and Linux systems. This would result in a situation in which text typed into a computer terminal, rather than composing an algorithm, would instead be directly rendered as raw sound, as used by ap/xxxxx in their performances.[7] The exact same text that in McLean's hands produced a highly danceable minimal techno would now be heard as seemingly random, and sometimes painful, noise.

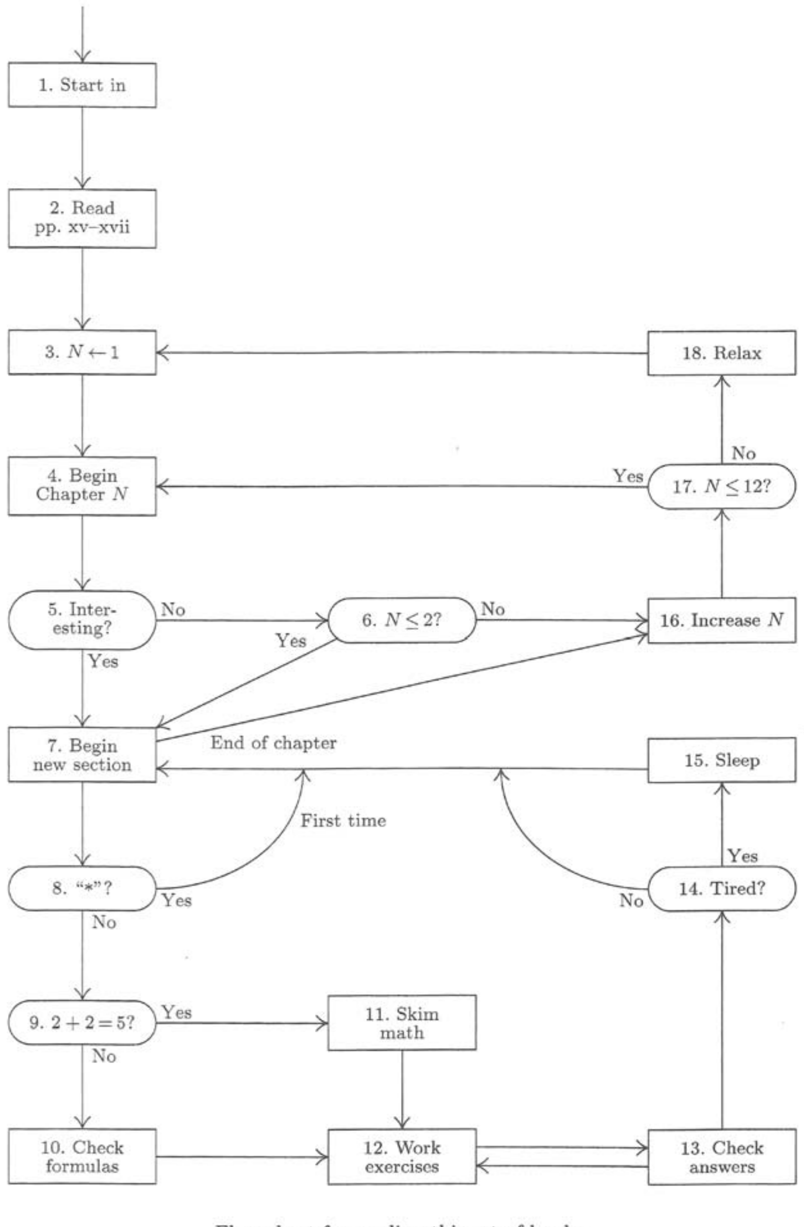
Another twist might take us in a different direction, like that of Amy Alexander's *extreme whitespace* which injects blank characters into the command line terminal as you type.[8] Also written in Perl and operating according to a similar principle of feedback between terminal input and background process as McLean's program, *extreme whitespace* quickly turns the normally stable environment of the terminal screen into a swirling vortex of characters reminiscent of early video art.[9] These works, of ap/xxxxx and Alexander, also exploit a situational incongruity in their delivery, they exhibit a behaviour we might not expect, in a form we might not predict. They do so, however, through a practice of coding that is less aligned with Knuthian elegance and more closely associated with those used in system hacking and debugging. These bring forth an entirely different set of pleasures, ones that turn from the dexterous to the sinister.

Perhaps, ironically, it is Knuth who best explains such sinister pleasures. In his essay 'The Errors of TEX', he outlines the process of creating what he calls 'torture tests':

> Instead of using a normal, large application to test a software system, I generally get best results by writing a test program that no sane user would ever think of writing. My test programs are intended to break the system, to push it to its extreme limits, to pile complication on complication, in ways that the system programmer could never have consciously anticipated. To prepare such test data, I get into the meanest, nastiest frame of mind that I can manage, and I write the cruellest code I can think of; then I turn around and embed that in even nastier constructions that are almost obscene. The resulting test program is so crazy that I couldn't possibly explain to anybody else what it is supposed to do; nobody else would care! [10]

It is significant how much of the 'torture test' embodies the inverse of Knuth's own paradigms of elegance, in particular that the torture code is so complex as to be unintelligible and impossible to explain. If elegance is dependent upon a certain adherence to or construction of norms, the performance of programming itself is dependent upon the cruel and obscene.[11] These monstrosities are the forms in which the materiality of computational process is convulsed into view, confronting us with disturbing presences that disrupt our normative expectations of how code should operate. For a norm to be effective it must demonstrate, must prove in performance, its ability to transcend such monstrosities. But there is little horror in Knuth's torture, for it seems that it is not so much that he seeks pain but laughter, that moment when physical composure becomes unravelled and erupts into the social.

Throughout his career, Knuth has emphasized that often seemingly pointless but pleasurable activities are an important aspect of how a programmer comes to develop their craft.[12] A recent publication, *Selected Papers on Fun and Games*, documents many of his own such activities. These include gathering photographs of various diamond-shaped road signs, collecting number plates with mathematical puns in them and his first published writing, the fictitious *Potrzebie System of Weights and Measures*, printed by his school journal and then *MAD* magazine in 1957.[13] Indeed, *The Art of Computer Programming* is littered



Flow chart for reading this set of books.

Figure 2.1 Flowchart from 'Procedure for Reading This Set of Books'

Source Knuth, Donald E., *The Art of Computer Programming, Volume 1: Fundamental Algorithms,* 3rd edn, p. xii–xiii. ©1998. Printed and electronically reproduced by permission of Pearson Education, Inc., Upper Saddle River, N J.

with various jokes, games and pranks. The first volume of the series opens with a section entitled 'Procedure for Reading This Set of Books' which is set out in the form of an algorithm accompanied by a flowchart. The reader is admonished to 'follow the steps faithfully', and these include instructions such as:

14. Are you tired? If not, go back to step 7.
15. Go to sleep. Then, wake up, and go back to step 7.[14]

For a book that is regarded by many as one of the defining texts of computing science, and that, for Knuth, constitutes a major life's work (he began writing in 1962 and is yet to complete all seven volumes), this is perhaps not what we might expect. While the style of humour may be particular to Knuth, the fact of humour being so integral to the practice of programming goes deeper than just one writer's authorial manner. Jokes, humour and the absurd have a relation to programming that is not only cultural but also constituent to its very being.

A symptom of professional immaturity

Released by MIT Press in 1978, Roger E. Kaufman's *A FORTRAN Coloring Book* is the first published computing text to use cartoon and comic strip drawings as a pedagogic medium.[15] Its approach has been adopted by a number of other texts on languages such as C, Pascal and Lisp, and, it could be argued, is the archetype to the entire *For Dummies* series and all its numerous imitators.[16] There is, however, something dark within Kaufman's text that sets it apart from the more anodyne humour of these later works. The *For Dummies* books primarily draw upon humour as a means to grease the wheel of cognitive capital, facilitating the ever-recurrent re-skilling (and de-skilling) of the contemporary IT worker. They represent an end-point in the transformation of the use of humour to aid production within the workplace, which has devolved from being a liberal characteristic of privileged workers such as scientists and creatives, as explored in Arthur Koestler's *The Act of Creation* (1964), to being a general form of managerial control known as 'structured fun'.[17] Stylistically, *A FORTRAN Coloring Book* most resembles the world of *Dr. Seuss*, and rather than offering a series of ironic platitudes such as stalk the pages of *For Dummies* books, it engages in a scatological satire that positions the reader in a somewhat Freudian relation to its subject. The opening pages describe the computer as 'like your Mommy's Bureau Drawers', illustrated as a piece of ornate Gothic furniture that looms over the accompanying text. Input/output routines are explained in reference to a part-human, part-bird-like character sat upon the toilet, and arrays are introduced through a short example called FONDLE that traverses the bureau drawers. For all its invocation of the format of children's literature, *A FORTRAN Coloring Book* is not a book for children. Its use of cartoons and

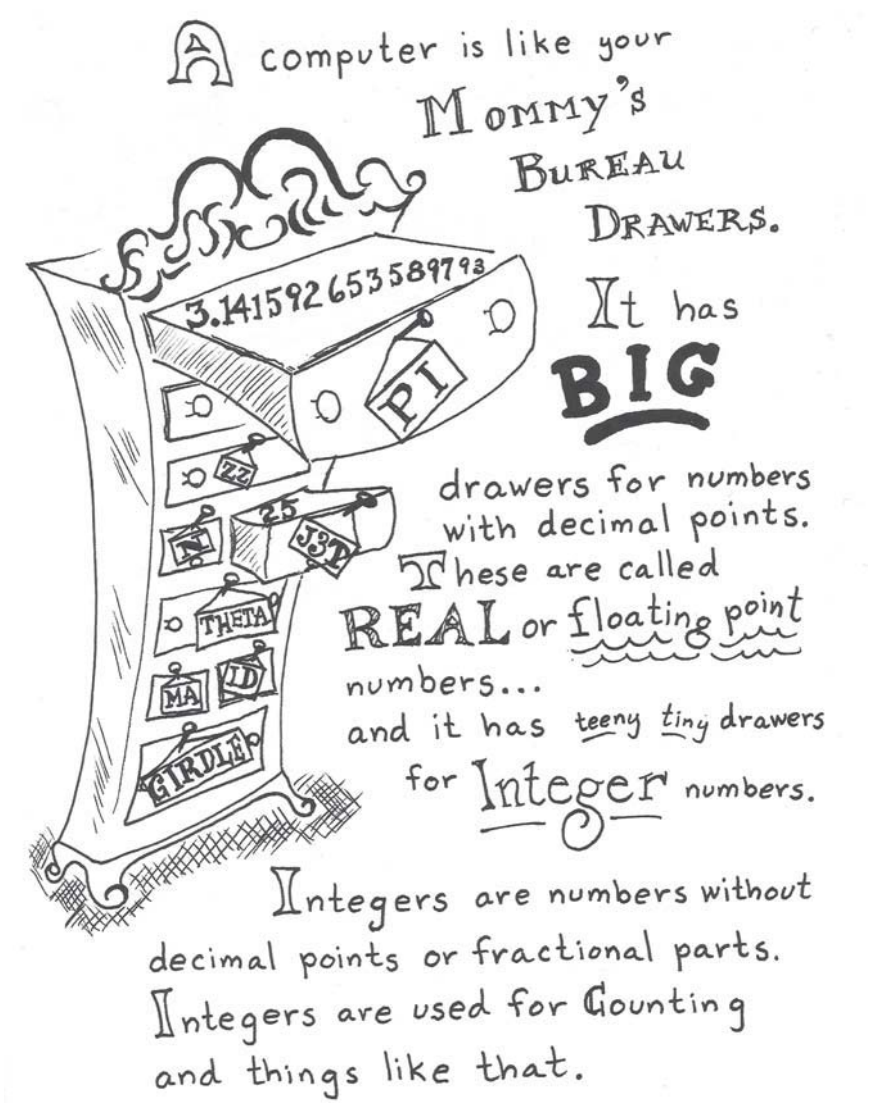

A computer is like your Mommy's Bureau Drawers.
It has BIG drawers for numbers with decimal points.
These are called REAL or floating point numbers...
and it has teeny tiny drawers for Integer numbers.
Integers are numbers without decimal points or fractional parts.
Integers are used for Counting and things like that.
3.14159265358979 3
PI
ZZ
25
N
J3T
THETA
MA
ID
GIRDLE

Figure 2.2 'A Computer is like Your Mommy's Bureau Drawers..'.

Source Kaufman, Roger, *A FORTRAN Coloring Book*, p.1, © 1978 Massachusetts Institute of Technology, by permission of the MIT Press.

the particular bite of its humour have more in common with the unofficial fanzine culture of graduate research communities. These publications, produced on Xerox machine and photo-stencil, were a staple part of environments such as the AI Labs at the Massachusetts Institute of Technology (MIT), and their humour can be seen in the various cartoons that adorned the pages of AI Lab memos, including the early EMACS manuals written by Richard Stallman.[18] Although based at George Washington University, Kaufman gave classes on FORTRAN at MIT, and it was for these that *A FORTRAN Coloring Book* was originally written.

In an essay of 1975, entitled 'How Do We Tell Truths that Might Hurt?' Edsger Dijkstra dismisses the use of anthropomorphisms in computing as 'a symptom of professional immaturity'.[19] While not addressed to Kaufman, such might be the description given to the animate bureau drawers and various hybrid creatures of the *Coloring Book*. From Dijkstra's perspective, this kind of heavily metaphorical presentation of programming merely causes confusion between what programming is and is not, between what it should and should not do. In responding to this critique, McLean has argued that 'metaphor necessarily structures our understanding of computation', and elaborates this through Alan Blackwell's analyses of different metaphors invoked by programmers, in which 'components were … described as actors with beliefs and intentions, being social entities acting as proxies for their developers'.[20] Blackwell's findings echo the encouragement that Knuth gave to students that in writing programming documentation they should find suitable metaphors in which to express how the code works, openly dismissing Dijkstra's position in doing so.[21] But the significance of Kaufman's book is not that it simply provides us with a set of engaging and entertaining metaphors through which we can learn to program. We can read *A FORTRAN Coloring Book* in ways other than those of an instruction manual. Its relation to *Dr. Seuss* is not merely one of shared drawing styles but also one of literary and narrative genres. As a literary work, *A FORTRAN Coloring Book* belongs to the tradition of absurdism and fantasy, of Bakhtinian carnival, that includes the inverse logics of *The Land of Cockaigne*, the hybrid creatures of medieval bestiaries, various works of Lewis Carroll and Douglas R. Hofstadter's *Gödel, Escher, Bach*.[22] As this summary of literary associations suggests, *A FORTRAN Coloring Book* is a work that should be read as an assemblage. In its internal structure it does not conform to a singular genre, it is not purely an instructional manual, nor is it a fully developed satire or, arguably, even that much of a colouring book, but rather as an emergent example of its own genre, it assembles different forms and figurations into itself. The book is also itself an element within a much larger assemblage of other texts, objects and practices. This is an assemblage that includes Kaufman's other writings, programs and engineering projects, but also those of others within the emergence of computer science and the much more circuitous co-evolution of modern humour, logic and computational practices.

Whereas Dijkstra and Knuth contributed to the development of computer science itself, Kaufman has been primarily involved in the application of software to engineering issues. Kaufman came to engineering through work in designing and building theatrical stage equipment, including early experiments in digital control systems.[23] One of his main contributions to computing was KINSYN (KINematic SYNthesis), originally developed in the late 1960s and 1970s, written in FORTRAN and running on an IBM 1130.[24] Developed out of Kaufman's theatre work, KINSYN is regarded as the first visual simulation tool for designing kinematic mechanisms.[25] Early applications of the tool included the design of an orthotic knee brace and, following from this, the development of assistive technologies has been an ongoing area of interest for Kaufman.[26] His other main book, *Introduction to Burmester Theory*, provides a theoretical and mathematical background to kinematic synthesis. It was originally produced in 1973 as a hand-written duplicate and featured cartoons similar to those in *A FORTRAN Coloring Book*.[27]

Like Dijkstra and Knuth, Kaufman began working at a time when much of the standardization and norms of programming languages and practices familiar to current-day programmers were not fully established. He had to implement his own FORTRAN compiler in order to support the calculation of complex numbers required for KINSYN, and in his initial work using SNOBOL (an early symbolic manipulation language developed at Bell Labs) he exposed numerous bugs and errors in the core language implementation.[28]

FORTRAN was developed at IBM in the 1950s, and derived its name from *The IBM Mathematical FORmula TRANslating System*.[29] The emphasis within the original language design was on providing a means of converting mathematical formulae into assembly code instructions. As such it retained many of the features of assembly language programming that explicitly reference the computer memory and hardware, and early versions did not provide the kinds of higher-level abstractions that simplify program structure such as recursion and sub-routines.[30] This also resulted in a lack of standardization across different hardware implementations, with certain routines, such as a DO loop, outputting different results depending on the particular machine they ran on.[31] Writing FORTRAN programs therefore required familiarity with the internal mechanisms of the computer, in order to understand issues such as memory allocation and addressing, output formatting, etc. However, as access to computers was limited during this period (prior to the rise of personal desktop systems), many programmers often had little hands-on experience with the machines they wrote for.[32] A program was typically written by hand on paper, known as a coding form, and typed onto a set of punch-cards which were then fed into the computer. To program, it could be said, was to compose a form of formalized literature directed to an imaginal machine.

The humour of Kaufman's book is endemic to this situation, not merely as a response to its practical frustrations and absurdities, but as a kind of theatrical praxis which mediates between the various manifestations of the human and machinic that constitute its assemblage: the encounter between mathematics, machines and aesthetics. This is based within the theatrical for it depends upon a 'bringing into view' of disparate forms and actions from which a theory of the computational might develop, yet which is not yet fully defined.

First published in 1900, *Le Rire: Essai sur la signification du comique* (Laughter: An Essay on the Meaning of the Comic) has been described as Bergson's only text to attempt a systematic treatment of aesthetics, primarily drawing upon dramatical forms of comedy such as Molière.[33] What is distinctive about Bergson's text is that it situates the humorous, the comedic, within a zone of contagion between the vitality of the human subject and the functional performativity of the automata, between spirit and matter:

> This soul imparts a portion of its winged lightness to the body it animates: the immateriality which thus passes into matter is what is called gracefulness. Matter, however, is obstinate and resists. It draws to itself the ever-alert activity of this higher principle, would fain convert it to its own inertia and cause it to revert to mere automatism. It would … imprint on the whole person such an attitude as to make it appear immersed and absorbed in the materiality of some mechanical occupation instead of ceaselessly renewing its vitality … Where matter thus succeeds in dulling the outward life of the soul, in petrifying its movements and thwarting its gracefulness, it achieves, at the expense of the body, an effect that is comic.[34]

Bergson's attitude towards automatism sits within his broader critique of the mechanical, which, in certain aspects, follows from that of Romantic thinkers such as Herder in positing an opposition between the mechanical and the organic, the material and spiritual.[35] The mechanical, as that which is an artificial assemblage of disparate elements not normally related to one another in 'nature', contrasts with the organic as that which is constituted from innately related elements forming an integrated whole in which each performs its 'naturally' given role or, in Bergson's conception, the integrity of their common duration.[36] The machinic has materiality and motion, but no spirit or grace, for its motion derives from the spatialization of time as number. This takes the mechanical as comprising both physical machines and institutional forms – the examples in *Le Rire* include both the operation of machines and the formalistic behaviour of bureaucrats. Bergson contends that the comic lies in the exposure of a repetition in behaviour which, like mathematics, exists outside the experience of time, contradicting the accrual of difference embodied in his notion of duration. However, while he touches upon the role of repetition in poetry, he does not make a clear distinction between repetition that is comical and that of, say, dance or music, which may similarly contradict our sense of time as duration yet be elegant, joyous or grave.[37] As his examples demonstrate, it is not the fact of repetition in itself from which the comical arises, but rather, that the repetition appears incongruous in regard to existing social norms. Humour here performs a normative role, it 'corrects men's manners' and differentiates society according to that which is desirable and undesirable within a given social order:

> Laughter must be something of this kind, a sort of *social gesture*. By the fear which it inspires, it restrains eccentricity, keeps constantly awake and in mutual contact certain activities of a secondary order which might retire into their shell and go to sleep, and, in short, softens down whatever the surface of the social body may retain of mechanical inelasticity. Laughter, then, does not belong to the province of esthetics alone, since unconsciously (and even immorally in many particular instances) it pursues a utilitarian aim of general improvement.[38]

Humour relates the aesthetic to the moral. That which becomes comical is that which fails the judgement of 'good sense' – Bergson's *Le Bon Sens* addresses issues similar to *Le Rire*.[39] This 'good sense' includes examples that today's reader might consider merely prejudicial, such as the discussions about the hunchback and Negro as objects of laughter. Humour and its morals are historically and culturally situated, and similar examples are to be found in Schopenhauer's treatment of humour in *The World as Will and Idea* (1819) and Lewis Carroll's *Symbolic Logic* (1896).[40] As an instrument of social differentiation, humour delineates the terrain of discrimination from which dominant groups and classes operate. It may also, however, facilitate practices through which various alternate assemblies seek to constitute themselves against that.

Eric Raymond's *The Jargon File* is a compendium of slang, peppered with various jokes and re-instrumentalizations of language through which the 'old school' hacker culture, in which Kaufman worked, identifies itself.[41] In her study of hacker culture, Gabriella Coleman defines humour as integral to hacking practice, not only as a form of social differentiation, but also as a modus operandi. Coleman argues that the structure of the hack, as the bringing together of unrelated elements so as to produce an unexpected functionality, parallels that of the joke, citing Mary Douglas's definitions of joking as a practice that combines 'disparate elements in such a way that one accepted pattern is challenged by the appearance of another'.[42] She illustrates this with an example of a succinct Perl hack that, in its attentiveness to the aesthetics of the language, is analogous to those of McLean's *feedback.pl* and Alexander's *extreme whitespace*:

```
#count the number of stars in the sky
$cnt = $sky =~ tr/*/*/;
```

The code counts the number of asterisks (stars) within a piece of text held by the variable named $sky compressing what might normally be six lines of code into one by exploiting 'certain side effects found in the constructs of the Perl language', and thereby creating a kind of poetical play within the source code of a program.[43]

In his theory of the joke as a 'diagram of innovative action' Paolo Virno describes the joke as an empirical use of language that tests the contingency of normative behaviour and grammatical constructs.[44] The joke is a playful use of language that performs unexpected twists of meaning, short-circuits of logic and, in so doing, disrupts both linguistic and social norms. In developing this theory, Virno adopts a set of terms from Aristotle's *Nicomachean Ethics*:

> a.) *phrónesis*, or practical know-how; b.) *orthós logos*, the discourse that enunciates the correct norm according to which the action in one single case takes shape; c.) the perception of *kairós*, of the proper moment for performing an action; d.) *éndoxa*, that is, the opinions prevalent from time to time within a community of speakers.[45]

In particular, the joke applies innovative, or unexpected, forms of know-how, *phrónesis*, in order to test the viability of given *éndoxa*, and thereby expose the contingency of 'all situations' and 'the way in which these situations are to be dealt with'.[46] As such, Virno's theory is entirely the opposite of Bergson's, in which the joke serves to establish and reinforce prevailing *éndoxa*.

Among his personal engineering projects, Kaufman demonstrates a hacker-inventiveness in his *PantsPutterOnner* and *ShirtPutterOnner* devices which, at first glance, have the appearance of Heath Robinson or Rube Goldberg contraptions, made, in one case, from elements that look to be adapted from household plumbing. These are a 'pair of dressing machines … designed and built … for a student born without arms'.[47] Their construction epitomizes the elegance of a hack that, taking existing materials and given existing norms, makes possible that which might otherwise not be. Susan Leigh Star has questioned the way in which technologies are brought to address such needs, as though it simply 'were a matter of expanding the exhaustive search for "special needs" until they are all tailored or customized'.[48] She calls instead for a questioning of the 'distribution of the conventional', asking, 'What is the phenomenology of encounters with conventions and standardized forms, as well as with new technologies?'[49] Unlike cosmetic prostheses, however, Kaufman's devices do not seek to standardize the body in conformance with societal norms (such as we would impose if we were to give the student prosthetic arms and insist he dress 'like everyone else') but rather accept and accompany the given physicality of their user. Seemingly arising from the joke and the very forms of that which is derogated in Bergson's 'utilitarian aim of general improvement', the automata and the 'deformed' body, the dressing machines instead demonstrate that even the most mundane of activities, that of getting oneself dressed, can bring into action numerous forms of *phrónesis* entirely outside existing *éndoxa*.

The *orthós logos* announced by Bergson, that it is laughable for a living body to incorporate the machinic (the becoming-machine), mirrors that of Dijkstra, that it is immature for the machinic to be given human qualities (the becoming-human).[50] What emerges between the elements assembled across Kaufman's practice, however, between the dressing machines and the colouring book, is neither simply mecha-morphic nor anthropomorphic, but rather that which 'affords opportunity for realising that an accepted pattern has no necessity'.[51] The hack/joke replicates the structure of the machinic. Each constitutes an assemblage of elements that are not 'normally' brought into relation – whether in terms of societal norms, linguistic standards or conventions of the 'natural'. Kaufman's work does not succumb to the criticisms of Bergson or Dijkstra, however, but rather exposes their contingency. Virno ascribes the 'violence' of human language to its potential to negate being, to say that a person is a not-human. In his critique of Hegelian dialectic, Bergson dismisses the category of the negative as the construction of a 'false problem' that constructs criteria that cannot exist, and yet laughter in *Le Rire* is itself the performance of such 'violent' negation.[52] Here, laughter responds to and identifies the not-human within the human, just as Knuth's torture test identifies the not-computational within his own software code. For Virno, however, humour can move beyond such first-order negations to not only expose the contingency of such patterns but also, as in Kaufman, to become the negation of negation.[53]

Kaufman constructs a comedic theatre that operates through a 'bringing into view' of all that which is positively regarded (that which constitutes 'theory', the thing we choose to contemplate) and that which is negated (the not-human, the not-computational) and presents them as an ad hoc assemblage that is prior to any such selection or normative structure. In Greek tragedy the body of the dead hero would be revealed upon the *ekkyklêma*, a piece of machinery wheeled onto the stage, exposing a truth that might otherwise be hidden.[54] In comedies the body would spring to life, or something entirely unexpected would appear. Kaufman's play brings on stage the machinery of computation and its concomitant desires and ambiguities. His characters perform the hybrid agency of human and non-human actors, the 'promises of monsters' of which Donna Haraway wrote: 'Their boundaries materialize in social interaction among humans and non-humans, including the machines and other instruments that mediate exchanges at crucial interfaces and that function as delegates for other actors' functions and purposes'.[55]

The promise is not entirely made good in Kaufman, however, for there are conflicting tendencies within his satire that echo the 'unequal structuring' between objects, science and nature whose problematic Haraway maps. Kaufman delights in the base corporeality of the computational in a form of Bakhtinian grotesque, yet, at the same time projects ambivalent anxieties through the construction of the computer as the mother-bureau hybrid, both desired and feared. This Oedipal configuration invokes the female-monster figures of eighteenth-century satire, such as Swift's Goddess Criticism and Spenser's Errour. These, as Susan Gubar has argued, allegorize what their authors considered to be forms of dangerous and unruly writing conflated with uncontrollable natural processes of endless eating, defecating and childbirth.[56]

This suggests a different reading of the anthropomorphic in regard to the computational, one that expresses both that which gives a human form to a non-human object but also that which shapes the human. The 'hurtful truth' pronounced by Dijkstra was directed against what he considered to be the false claims of John von Neumann that human brains were a form of computing machine that a computer could mimic and the ideological ambitions of AI.[57] Dijkstra's critique can be understood as an accompaniment to the distrust of machinery that haunts his other writings, and in response to which he enacted his own anthropomorphism. Dijkstra performed computation itself as an inherently human action, insisting on working out programs in long-hand writing rather than on a machine (long after coding forms became obsolete) and arguing that the true computer for which the programmer wrote was not the compromised physicality of hardware but the pure intellectual 'machine' of the programming manual:

> Eventually, there are two 'machines'. On the one hand there is the physical machine that is installed in the computer room, can go wrong, requires power, air conditioning, and maintenance and is shown to visitors. On the other hand there is the abstract machine as defined in the manual, the 'thinkable' machine for which the programmer programs and with respect to which the question of program correctness is settled.... Originally I viewed it as the function of the abstract machine to provide a truthful picture of the physical reality. Later, however, I learned to consider the abstract machine as the 'true' one, because that is the only one we can 'think' ...[58]

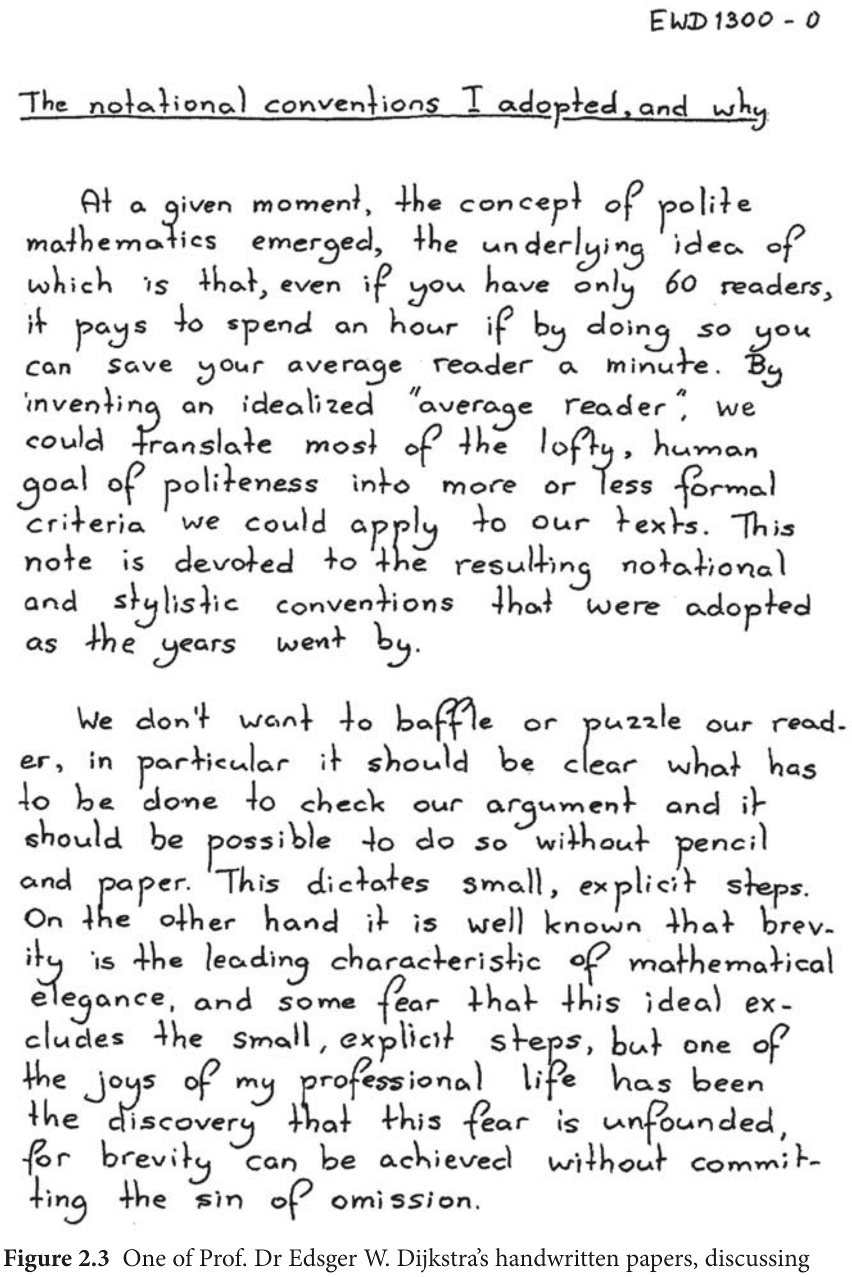
There is an irony in Dijkstra's practice for it is almost as though he were unable to escape the *mise-en-scène* of Turing's description of the problem of computation as that of a human 'computer' sitting at a desk writing and erasing marks on an endless strip of paper, not knowing when, or if, he can stop.[59] There is a double irony in that Dijkstra pioneered the interrupt mechanism, an element relating external hardware, such as keyboards, to the internal processing system. The interrupt enables a programmer to write and run code interactively on a computer, a key innovation that rendered the coding form obsolete, and without which McLean's *feedback.pl* would not be possible.[60] Dijkstra's distrust also echoes that of many mathematicians towards computer-demonstrated proofs, such as evidenced, famously, in the lukewarm response to Appel and Haken's proof for the Four Colour Theorem (1976). The theorem seeks to determine whether or not all the countries on a map can be coloured in using only four distinct colours and ensuring that no two neighbouring countries are coloured the same. This was the first major theorem to be proven using software assistance having eluded purely human analysis since it was originally conjectured in 1852, yet its solution was not celebrated.[61] The negative response of the mathematics community is epitomized in the words of one critic, Ian Stewart, who argued that this approach did not explain *why* the proof was correct:

> This is partly because the proof is so long that it is hard to grasp (including the computer calculations, impossible!), but mostly because it is so apparently structureless. The answer appears as a kind of monstrous coincidence. Why is there an unavoidable set of reducible configurations? The best answer at the present time is: there just is. The proof: here it is, see for yourself. The mathematician's search for hidden structure, his pattern-binding urge, is frustrated.[62]

The Appel-Haken proof confounded the aesthetics of mathematicians for the set of 1,482 different map configurations required to verify it could not be grasped by human imagination.[63] It was not succinct. It was not elegant. In combining human and mechanical means it transgressed the prohibition against crossing between different disciplines such as that between arithmetic and geometry, the *metabasis ex allo genos*, as established in Aristotle's *Posterior Analytics*.[64] The proof exuded an excess of computational materiality, it had contagion, it could not be given human shape.

Dijkstra's long-hand computations can be fully understood as anthropomorphic in the dual sense of that which gives human shape to something, and as that which shapes the human. Elsewhere in the essay critiquing anthropomorphism Dijkstra argues: 'The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities'.[65]

The choice between computational machine on the one hand and pen and paper on the other is, for Dijkstra, a conscious choice of how the programmer himself is shaped.[66] The expression of such a choice is also the projection of a particular becoming-human onto the tool itself, and therefore a form of anthropomorphism. As Paul de Man writes, the anthropomorphic is that which seeks 'to reconcile the pleasures of the mind with those of the senses and to unite aesthetics with epistemology'.[67] Barany and MacKenzie argue that the persistence of chalk and blackboard within mathematical practice, long after digital displays and PowerPoint have dominated all other areas of academic presentation, is that the physical writing of the equations constitutes a form of performance through which mathematical proofs are given material validity.[68] It is as though the equation must be written out, rather than simply shown on a slide, in order for it to come into being as an object that we can perceive (aesthetics) and comprehend (epistemology). When aesthetics and epistemology fail to coincide the effect may either be monstrous or comic.



EWD1300 - 0

The notational conventions I adopted, and why

At a given moment, the concept of polite mathematics emerged, the underlying idea of which is that, even if you have only 60 readers, it pays to spend an hour if by doing so you can save your average reader a minute. By inventing an idealized "average reader", we could translate most of the lofty, human goal of politeness into more or less formal criteria we could apply to our texts. This note is devoted to the resulting notational and stylistic conventions that were adopted as the years went by.

We don't want to baffle or puzzle our reader, in particular it should be clear what has to be done to check our argument and it should be possible to do so without pencil and paper. This dictates small, explicit steps. On the other hand it is well known that brevity is the leading characteristic of mathematical elegance, and some fear that this ideal excludes the small, explicit steps, but one of the joys of my professional life has been the discovery that this fear is unfounded, for brevity can be achieved without committing the sin of omission.

Figure 2.3 One of Prof. Dr Edsger W. Dijkstra's handwritten papers, discussing 'polite' notation, EWD1300

Source Full text: http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1300.PDF

Selfish computers of happiness

Knuth's 'Computer Programming as an Art' was originally presented as his acceptance speech for the 1974 Turing Award.[69] It takes the form of a survey of different discussions about art and aesthetics in relation to computation and number with references that range from a definition of logic by thirteenth-century philosopher Duns Scotus to a nineteenth-century manual on accountancy. Within this there are two philosophers whom he quotes at length and gives particular prominence to: Jeremy Bentham and John Stuart Mill.[70] From Bentham he derives the concept of utility as the best measure of good and bad: 'There is no taste which deserves the epithet good, unless it be the taste for such employments which, to the pleasure actually produced by them, conjoin some contingent or future utility…'[71]

Bentham's concept of the good as that which is both measurable and pleasurable, and given algorithmic expression through his felicific calculus, underlies and gives a moral dimension to Knuth's concept of programming elegance and its more pragmatic expression, explored at some length in *The Art of Computer Programming* and other texts, of seeking to quantitatively measure the scale and efficiency of a given algorithm in terms of its running time and use of resources.[72] From Mill he derives the argument that art and science are complementary, that all disciplines have both an art and a science to them, with art supplying the capacity to judge that knowledge produced by science which has greatest value to practice: 'Art … brings together from parts of the field of science most remote from one another, the truths relating to the production of the different and heterogeneous conditions necessary to each effect which the exigencies of practical life require'.[73]

Here, art is that which defines the *conditions* under which knowledge becomes productive. For Knuth these conditions are both *economic*, the effective use of resources, and *aesthetic*, a sense for that which is harmonious and 'good'. Bentham and Mill help define what may be described as Knuth's ethics of production, the behaviours and moral values under which programming is practised. The realization of this in elegant code gives material shape to the larger project of Literate Programming which Knuth has advocated throughout his career and created software tools to facilitate. Just as these ethics draw upon the ideas of English liberal philosophers, Bentham and Mill, the practice of Literate Programming also adopts the aesthetic medium most closely related to the expression of liberal thinking, the essay: 'The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style'.[74]

For the Earl of Shaftesbury the essay was the ideal vehicle through which to define and disseminate the new subjectivity of the self-possessing liberal individual.[75] This was given form through Shaftesbury's concept of *politeness* as a mode of cultural and political action. Acquired through appropriate education and demonstrated through exhibiting appropriate taste, politeness offered a means of acquiring authority within the transactions of daily life and commerce that did not derive from the institutional power of the monarchy or the Church whose influence had fallen in the wake of the English 1688 Revolution. Depending as it did upon access to a privileged means of education and consumption, the practice of politeness retained social differentiations while being able to operate across all contexts, assuring both liberty and distinction to those who had it. Politeness was both a literary and theatrical practice for it was disseminated through particular modes of writing, and applied as a kind of extemporary performance on the social stage. This gave voice to a moral philosophy that was conversational and concrete rather than speculative and overly abstract, and whose aesthetic was its means of appeal: 'The author wrote in an agreeable English. He punctuated the discourse with humour. He preserved moral scale, eschewing, at one end of the spectrum, excess detail and elaboration and, on the other, mysterious depth and abstraction'.[76]

While politeness gave qualitative polish to the emerging discourse of English liberalism, it was conjoined with another mode of writing that gave it an effect of quantitative rigour, that of mercantile and scientific numerical writing. This is described by Mary Poovey as a *gestural mathematics*, the use of mathematical language to describe philosophical concepts and thereby suggest a correlation between the methods of mathematical proof and philosophical argument.[77] Rather than defining the aesthetic as an opening to pure sensuality (as the Romantics would later do), the mathematical precedes and shapes the aesthetic as defined in this era, for it is from the potential of mathematical thought to consider things in the abstract that the notion of the disinterested contemplation of the aesthetic derives. Similarly, the concept of the beautiful derives from analogy to the harmony of mathematical proportions. This abstract basis of aesthetic judgement appears to run counter to the tangibility of discourse promoted by Shaftesbury. For Shaftesbury, however, this is resolved in terms of class. The aesthetic is not that which is abstract from the world but rather from labour, for those who work with their hands, absorbed in materiality through their labour, are unable to think in terms of either the abstract or the disinterested; they are, in a word, unaesthetic.[78] Aesthetics here is a moral-political practice, it makes the abstract concrete and gives it *value*, for it seeks to instil behaviour that is proportionate and 'true'.

Humour is integral to the realization of this. In 'Sensus Communis; an Essay on the Freedom of Wit and Humour' (1709), Shaftesbury argues that irony and satire (what he calls 'raillery') are an ideal means of testing the logic and substance of debate:

> They may perhaps be Monsters, and not Divinitys, or Sacred Truths, which are kept thus choicely, in some dark Corner of our Minds: The Specters may impose on us, whilst we refuse to turn 'em every way, and view their Shapes and Complexions in every light. For that which can be shewn only in a certain Light, is questionable. Truth, 'tis suppos'd, may bear all Lights: and one of those principal Lights or natural Mediums, by which Things are to be view'd, in order to a thorow Recognition, is Ridicule it-self, or that Manner of Proof by which we discern whatever is liable to just Raillery in any Subject.[79]

In subjecting ideas to the test of ridicule, humour acts as an 'instrument of reason' exposing that which claims to be proportionate and true yet which rests upon a logic that is deformed and ugly.[80] As in Knuth's torture test, raillery debugs theory.

For the second edition of *Characteristicks* of 1714 Shaftesbury commissioned a series of engraved iconographies to illustrate its subject matter from the artist Simon Gribelin and worked closely with the printer, John Darby Jr, to produce the publication to exact standards ensuring that no ink showed through the pages as had happened with the first edition.[81] Dismayed by the poor quality of the electronic typesetting of the second edition of *The Art of Computer Programming*, Knuth set about writing his own software to typeset his books. Intended as a short sabbatical project, the work stretched to more than ten years and culminated in TEX, now one of the most widely used digital publishing tools.[82] The first book to be fully typeset in TEX was *Concrete Mathematics* (1989). Written in collaboration with Ronald L. Graham and Oren Patashnik, the book grew out of a course taught at Stanford on the mathematical analysis of algorithms, started by Knuth in 1970 and first set out in the opening chapter of *The Art of Computer Programming*. A new font, called AMS Euler, was specially commissioned from designer Hermann Zapf to typeset the mathematics in the book. Echoing Dijkstra's preference for writing on paper, the font was designed to 'capture the flavour of mathematics as it might be written by a mathematician with excellent handwriting. A handwritten rather than mechanical style is appropriate because people generally create mathematics with pen, pencil, or chalk'.[83]

This explicit evocation of the handwritten can be read in relation to the authors' statement that the book is 'a kind of manifesto about our favourite way to do mathematics'.[84] A manifesto that is also expressed in the book's title, *Concrete Mathematics*, emphasizing the practical applicability of what it taught and consciously rejecting the Abstract Mathematics then in vogue in US universities and the New Math educational systems.[85] The book also evokes another entirely different practice of inscription. Running throughout the margins of the book are various jokes and comments gathered from students on the course that the authors call 'graffiti':

> Some of these marginal markings are merely corny, some are profound; some of them warn about ambiguities or obscurities, others are typical comments made by wise guys in the back row; some are positive, some are negative, some are zero. But they all are real indications of feelings that should make the text material easier to assimilate.[86]

This might seem a little idiosyncratic and self-indulgent, yet, in the light of the role of humour in Shaftesbury's doctrine of politeness and the authors' own explanation, it can be understood as integral to Knuth's ethics of production. It is as though the authors, in a display of polite self-deprecation, wish to demonstrate that the subject matter has been subject to the 'raillery' of the classroom. It also points towards a possible awareness on the authors' part of the role of humour in establishing and cultivating, what might be called, the normative pleasures of their practice. This connects to Bentham, in relating the intrinsic 'good' within a practice to the investment of pleasure and also to Knuth's support for recreational mathematics and toy problems as a means for developing that practice through disinterested, pleasurable activities. As Stanley Grean puts it in his study of Shaftesbury, humour here is 'both a method and a state of mind'.[87] Whereas, however, the Shaftesburian text gestures towards the mathematical so as to construct a sense of legitimizing order beneath its discourse, the graffiti of *Concrete Mathematics* and the jokes that punctuate *The Art of Computer Programming* are a form 'gestural comedics' whose role is not that they entertain or make us laugh, but rather that they correlate the subject matter of the texts to a particular temperament and sensitivity. It is through the gesture of the comedic that the graffiti construct and reinforce a particular *éndoxa*, 'the opinions prevalent from time to time within a community of speakers'. Literate Programming, therefore, is not simply a method for writing programs, but also of making programmers, of shaping subjectivities. Humour constructs a kind of class ethics. As though to paraphrase Shaftesbury: 'For without Wit and Humour, software can hardly have its programmers or be programmed'.[88]

Both Knuth and Kaufman reflect upon the teaching situation through their humour, but whereas for Kaufman this negotiates the jointly corporeal and imaginal relation between human and machine, in Knuth it negotiates the roles of student and tutor. The pedagogical context for Knuth is not simply one phase within that process but rather intrinsic to programming itself. Knuth describes the act of writing programs as like teaching somebody else to do a task. The programmer teaches the machine. The machine, however, also teaches the programmer through the way it enforces greater precision on the 'tutor' than a purely human teaching scenario might require.[89] The teaching situation was a recursive one, and this recursion gives rise to a new social formation: 'I thought about how it would be to live with such a machine and with the new tools that I was creating – sort of like living in a new subculture'.[90]

Like the Shaftesburian essay, Literate Programming teaches one how to behave in a new social order. Knuth's humour sits within the genealogy of polite liberal humour that stretches from Shaftesbury to Bergson for it serves to inform the normative conditions under which such a social and cultural order comes into being, a 'social gesture' that 'pursues a utilitarian aim of general improvement'. It does so through the construction of an ethics of production which is also a 'class' ethic, that, as explored in the work on Gabriel Tarde, links the schoolroom to the economic structure.[91] Elegant code, in this context, articulates the identity of the programmer within a particular professional class, equivalent to the English liberal conception of the 'order of ranks' and the French professional *cadres*.

While Shaftesbury was a leading influence upon and advocate of Whiggism, he was in certain respects reactionary in regard to his times and the later radical liberalism of Bentham. Looking back to the older concepts of virtuous republicanism of the pre-revolutionary writers he criticized those who embraced the explicitly quantitative values of emerging mercantilist society as 'selfish Computers of Happiness'.[92] Despite the references to Bentham, Knuth does not give over to a fully economic utilitarian conception of computing under the demands of *homo* *oeconomicus*, perhaps owing something to a belief in the principle of academic work as a shared public good. The full economic and political consequences of a return to Bentham in twentieth-century computing do not arise from Knuth's work. It is within von Neumann and Oskar Morgenstern's *Theory of Games and Economic Behaviour* (1944) that the selfish computation of happiness as economic utility finds a new form. While drawing on the more rigorous mathematical models of Daniel Bernoulli in implementing their system, they directly invoke Bentham's historical and moral arguments in justifying their approach.[93]

The instrument of humour has a more explicitly 'ideological' sense, however, in how *Concrete Mathematics* and *The Art of Computer Programming* position themselves in regard to other manifestos and philosophies within twentieth-century mathematical practice. Across the pages of *The Art of Computer Programming*, there lurks a spectre, making itself known only through the presence of a sinister sign:

It appears on the margins of the text to warn the reader of passages which may be of particular difficulty and require greater attention or caution. The sign takes its form from the roadsign symbol for a dangerous bend. While this may appear to be derived from Knuth's hobby of collecting diamond-shaped roadsigns, the origins of its use lie not within the US highway code but elsewhere within mathematics. The symbol was first introduced in *Éléments de mathématique* (Elements of Mathematics) by Nicolas Bourbaki, the first volume of which was published in 1939. While Knuth's writing style may follow the Shaftesburian tradition, the form and ambition of *The Art of Computer Programming* most clearly follows that of *Éléments de mathématique*. Both are multi-volume works which seek to provide a near definitive account of their discipline. Both, as a series, remain unfinished. Both works gradually build from basic first principles, set theory in Bourbaki, data structures and algorithms in Knuth, towards more complex domains. Distinctively, and unusually for mathematical and computing texts, both combine their theoretical expositions with historical notes outlining prior developments. And, prior to Knuth, it is in Bourbaki that we find the first acknowledgement that recreational mathematics has been a significant source of innovation and discovery.[94] Similarly, both Knuth and Bourbaki's works attempted to standardize mathematical notation and introduced their own symbols. Bourbaki introducing not only the dangerous bend symbol but also, more significantly, the sign for an empty set, $\varnothing$, which became a fundamental building block to their theory of number.[95] Knuth introduced the up-arrow symbol, ↑, for notating large integers, and sets out in the introduction to *The Art of Computer Programming* distinct uses of notation that separate the description of an algorithm from its mathematical analysis.[96] Typically for Knuth, the introduction of notation is not without humour, and at one point in his discussion he declares that '[t]he notations we are using in this section are a little undignified. … Our notations are almost universally used in recreational mathematics (and some crank literature!) and they are rapidly coming into wider use'.[97]

Despite these similarities Bourbaki is not cited in Knuth's major works. The formalized structural text of Bourbaki is the antithesis of the convivial elegance of Knuth. More fundamentally, Bourbaki was a key influence on the emergence of New Math teaching and the dominance of Abstract Mathematics against which *Concrete Mathematics* positioned itself.

Not only does the distinction between Knuth and Bourbaki relate to that of two opposing ways of doing mathematics but they each embody two opposing models of the construction of subjectivity and the role of humour within that. With Knuth humour coheres the subject, drawing upon the practices of subject formation from the rational liberal tradition. A private individuated 'self' that comes into being through its display of established normative patterns that are, in part, demonstrated through the humour that it performs. With Bourbaki, conversely, humour performs the very undoing of such a 'self'. There has never been a single, coherent Nicolas Bourbaki as a self who authored a series of mathematical texts, rather Nicolas Bourbaki is a syndicate of mathematicians. The name itself was, according to one account, originally given to a visiting speaker who would present the annual lecture to first-year students of the *École Normale Supérieure* in Paris. The lecture was a fiction, a prank, performed by an actor, the name chosen at random.[98] When a group of mathematicians connected with the *École* began working on a project to provide a comprehensive theory of contemporary mathematics, they adopted the name as their *nom de plume*. The group included, among many others, Henri Cartan, Jean Dieudonné and André Weil. Weil's description of their collectivized authorship presents a kind of literary practice that could not be further removed from the explicit personalization within Shaftesburian essay technique:

> For each topic, a writer was designated after a preliminary report and group discussion. This writer provided a first draft which the group would read and discuss again, modifying it to varying degrees or even, as happened more than

> once, rejecting it out of hand. Another writer would be designated to come up with a second draft, following the directives of the group – which of course were not always heeded; and so on.[99]

The prank was taken on by the mathematics community, many of whom continue to refer to Bourbaki as a singular individual.[100] The use of a syndicated identity was later adopted by other mathematicians, such as the Situationist-style maths fanzine *Manifold*, literary collectives such as OuLiPo, which included mathematicians and sought to base a new practice of writing on Bourbaki's axiomatic system, and appears in multi-use names such as Luther Blissett and Karen Eliot.[101] In the 1940s Bourbaki's attempt to join the American Mathematical Society (AMS) as an individual member was rejected by the secretary Ralph Boas who published an article denouncing Bourbaki as a group. In a manner prefiguring the deliberately provoked feuds of Stewart Home and the Neoists, Bourbaki responded with a letter refuting the allegations and claiming that Boas was a fictitious name under which a group of US mathematicians published the *Mathematical Reviews*.[102] Boas, it so happens, was also on the committee who commissioned Zapf to design the AMS Euler font, a font which in its evocation of a handwritten style foregrounds the idea of mathematics as the pursuit of self-contained subjects who are coterminous with their physical bodies. In contrast to Bourbaki's use of a common syndicated identity to recognize the collective endeavour of their mathematics, the authors of *Concrete Mathematics* explicitly named individual contributors.[103]

While Dijkstra's insistence on writing by hand might demonstrate a similar commitment to the coterminous self embodied in AMS Euler, his specific choices in writing machinery indicate a more complex, multiple performance of the self within his work. Among Dijkstra's self-published writings there are some that are typed out on letterheaded paper rather than handwritten. The letterhead is for Mathematics Inc., an international corporation, the chairman of which is none other than Dijkstra himself. This is not Dijkstra as scientist and mathematician but rather his own nefarious alter ego. The letters are included within the indexing system Dijkstra applied to his other scientific papers and included in an anthology of his work published by Springer in 1982, thereby giving them equal importance to his mathematical and computing work.[104] These are not scientific notes, however, but a satirical parody of modern management techniques and the commercial exploitation of mathematical knowledge. Central to Mathematics Inc. strategy is the ruthless pursuit of legal ownership and patent rights over mathematical theorems. Written in the wake of the establishment in 1967 of the World Intellectual Property Organization (WIPO), they can be seen as part of the longer struggle over Intellectual Property in academia and the later rise of copyleft and Free Software. In this regard, they have something in common with the authorial devices of Luther Blissett and Karen Eliot. Dijkstra's split self is not, however, the embracing of the syndicated identity of Bourbaki or Blissett but more of a Scriblerian attack against the influence of commercialization on academic research.[105] In one Mathematics Inc. piece, which is prescient of current directions in the corporatization of the university, the company announces a new product, the Mathematical Articles Evaluation System (MAES®), designed to automate the grading of mathematical papers and thereby undermine the autonomy of academic practice: 'the wholesale introduction of MAES® will teach the reactionary bastards! At last their private, hobbyist norms will evaporate, for MAES® will force them to adopt the standards of the mathematical industry'.[106]

MAES embodies an entirely different literary form, neither the privileged personal expression of the Shaftesburian essay, nor the syndicalized commonality of Bourbaki, but rather the corporatized, branded voice of *homo oeconomicus*. The algorithmic authorship executed by MAES restructures all thought under a singular legal regime rather than the debates of mathematical discourse.

Even as a joke isn't it evidently mathematics?

One of Dijkstra's main contributions to computing was his involvement in the design and implementation of the ALGOL programming language. ALGOL (from *ALGOrithmic Language*) was designed as a universal programming language which would not only work uniformly across different types of hardware but also provide a clear written presentation of an algorithm suitable for printing in scientific reports and journals. As such, it was intended both as a response to the inefficiencies and limitations of FORTRAN and, like Knuth's later development of Literate Programming, to support the dissemination of computing knowledge and pedagogics.[107] Whereas Literate Programming would adopt the use of an essay form wrapped around the programming code, in ALGOL the syntax and typographic conventions of the language itself were intended to provide sufficient clarity so that an algorithm presented in ALGOL could be read as a self-contained expression. This foregrounds the structure of the algorithm itself rather than the machine-specific descriptions of memory registers and allocations that were typical of FORTRAN code. ALGOL describes what the algorithm will do, while FORTRAN describes what the machine will do. In this way algorithms could be published in a written form analogous to that of the publication of a mathematical proof, rather than, as with FORTRAN, something akin to engineering notes. Unlike previous programming languages, which were often devised to suit a particular task, ALGOL was designed as a set of basic generalized axioms from which more complex statements could be derived. As such, it follows the work of Hilbert's project, as outlined in 'The Foundations of Mathematics' (1927), of defining mathematical practice in terms of how it can be expressed through a rigorous formalized language:

> For this formula game is carried out according to certain definite rules, in which the *technique of our thinking* is expressed. … The fundamental idea of my proof theory is none other than to describe the activity of our understanding, to make a protocol of the rules according to which our thinking actually proceeds.[108]

The emphasis upon formalization is reinforced in a later comment by Dijkstra in which he argues in favour of the phrase 'programming notation' rather than 'programming language':

> The introduction of the term 'language' in connection with notation techniques for programs has been a mixed blessing. On the one hand it has been very helpful in as far as existing linguistic theory now provided a natural framework and an established terminology ('grammar', 'syntax', 'semantics', etc.) for discussion. On the other hand we must observe that the analogy with (now so-called!) 'natural languages' has been very misleading, because natural languages, non-formalized as they are, derive both their weakness and their power from their vagueness and imprecision.[109]

Although Dijkstra was as strong an advocate of elegance as Knuth, this comment indicates something of a distinction in how each understood this.[110] While both might place emphasis upon the precise, efficient expression of an idea in code, for Knuth this has a rhetorical dimension in that code, as essay, should aim to be persuasive in expression and display an appropriate conduct on the part of the programmer – which can be contrasted with the 'obscene' and 'crazy' code of his torture test programs. For Dijkstra elegance lies more in an irrefutably self-evident correctness, for truly elegant code would not require commentary nor debugging.[111] For Knuth elegance is the start of a conversation, for Dijkstra it is the conclusion. While Knuth cites Bentham and Mill in defence of his ideas, the opening pages of the *Report on the Algorithmic Language ALGOL 60* quote the *Tractatus Logico-Philosophicus* (1921) of Wittgenstein: 'Was sich überhaupt sagen läßt, läßt sich klar sagen; und wovon man nicht reden kann, darüber muß man schweigen'.[112] (What can be said at all can be said clearly; and about that of which one cannot speak, one must stay silent.)

There is a sardonic humour in the use of Wittgenstein here that plays a similar role to the student graffiti in *Concrete Mathematics* in that it provides a commentary upon the text and the conditions under which it acquires meaning. In this regard, however, the humour is perhaps more profound than Dijkstra may have intended.

The inclusion of the quote is something of a rebuke to existing practices in computing and draws a modest analogy between ALGOL and the aims of Wittgenstein's *Tractatus* which seeks to define the limit conditions under which language might operate logically. The design of a programming language is, for Dijkstra, like a form of applied analytic philosophy seeking to clarify how we make use of the language. As the requirements set out in the ALGOL 60 report make clear, much of the process of design is one of defining limits. The language will only use basic alpha-numeric notations that do not privilege any particular mathematical method or symbolism (in this regard responding to problems in Kenneth Iverson's APL)[113] and can be easily reproduced in print (a response to the costs and complexity of mathematical typesetting that Knuth was to address differently in TEX). The language will use a minimal set of dedicated instruction symbols (such as =:, +, – and words such as begin, if, etc.) whose usage cannot be altered. It will place few constraints on how a programmer can create new words beyond this (a response to the restrictions that FORTRAN placed on naming variables). The language will not include any operations that are restricted to specific forms of hardware. As, at this time, the methods of printing, displaying and outputting information varied from one computing system to the next, this resulted in ALGOL having no defined way of outputting results. While Dijkstra, and the development team, considered this both necessary and appropriate (the report, echoing Wittgenstein, states: 'On this matter, the official ALGOL 60 report is as silent as the grave, and with very good reason')[114] the lack of any output system proved to be a major obstacle to the adoption of ALGOL, resulting in FORTRAN continuing in widespread use well into the 1980s, long after ALGOL itself was replaced by languages such as Pascal and C. Dijkstra's preference for the term 'programming notations' was similarly a demarcation of limits, indicating that there were many forms of expression that computing machines were not capable of, and encouraging a particular discipline of thinking within the programmer.

It has been argued that Dijkstra was one of the few computing scientists of this period who were familiar with the mathematical legacy within which Turing had worked, and his emphasis upon conceiving programs and computing systems as mathematical notations in terms of the axiomatic principles of Hilbert was at this point a controversial exception to the norm that favoured more of an engineering approach.[115] In an unpublished manuscript from the 1930s Turing discusses the problems of current mathematical notation and the need for standardization and reform that echoes the contemporary work of Bourbaki.[116] Dijkstra would not have known of this paper, but there are strong parallels in their concerns about the influence of notation systems on the performance of mathematical thinking. Like Dijkstra, Turing looks to Wittgenstein, citing a lecture given by Wittgenstein on mathematics that he had attended.[117]

These lectures were, however, to challenge the work of Hilbert and Wittgenstein's mentor Bertrand Russell. For Turing they may well have responded to his own criticisms of what he described as an 'extreme Hilbertian' perspective.[118] In these and in related notes from the late 1930s and early 1940s Wittgenstein began to question whether mathematics could have any kind of stable foundation of the kind sought by Hilbert. His approach was not to enter into the specific arguments of pure mathematics, to resolve or sustain particular paradoxes, but rather to discuss the process through which a thinking of mathematics takes place. This entailed a questioning as to what extent this relied upon contingent phenomena and factors that were external to mathematics itself. Wittgenstein argues that mathematics as a pure self-contained discourse is without sense, for it has no relation to an outside world. Mathematics is without meaning (sense as understanding) because it cannot be apprehended sensorially (sense as perception, aesthesis). Hilbert had declared that any object or entity could be used as a mathematical symbol, and Bourbaki characterized mathematical language as a stripping out of any pre-existing meaning from the words it used.[119] Pure mathematics could not be anything other than non-sense.[120]

Throughout the notes and lectures Wittgenstein explores various scenarios in which a calculation is either expressed or performed – as a spoken instruction, in the form of a notation, or through the action of a machine – showing that in each instance the mathematical only becomes known through external, contingent factors that influence our understanding of it. He similarly argues that norms and conventions within the discipline of mathematics change over time and that what has been accepted as 'correct' mathematics at one point may later be rejected as false, as Russell had demonstrated in exposing the paradox within Frege. Rather than seeking an ultimate correctness, the issue for Wittgenstein was that if a statement which was accepted as true in one system was shown to be false, or unnecessary in another, to what extent did it retain the identity of mathematical knowledge, was false mathematical knowledge still a part of mathematics?[121] Mathematics, for Wittgenstein, is neither the discovery of immutable pre-existing forms as in Platonist conceptions, nor the expression of an innate numbering capacity as in Intuitionism, but rather an activity of adopting and following particular rules.[122] As such he follows Hilbert's project to its conclusion but demonstrates, in doing so, that this offers no irrefutable foundation but rather an infinite regression of rules to follow rules which may lead one down unexpected twists and turns of logic and semantics. Written contemporaneously with Bourbaki's first publications, there is a comment in Wittgenstein's notes that can be read as drawing a connection through this between the raillery of Shaftesbury and the prankish origins of Bourbaki:

> Imagine set theory's having been invented by a satirist as a kind of parody on mathematics. – Later a reasonable meaning was seen in it and it was incorporated into mathematics. (For if one person can see it as a paradise of mathematicians, why should not another see it as a joke?)
>
> The question is: even as a joke isn't it evidently mathematics?[123]

Wittgenstein's observation, however, is not the acknowledgement of the role of raillery, satire or recreational mathematics on shaping 'serious' mathematical thought, but rather relates to Virno's conception of the joke as a means of exposing the contingency of prevalent norms. To think of mathematics simultaneously as a 'paradise' (invoking Hilbert's celebration of Cantor's theory of infinite number) and as a joke is to think mathematics as *necessarily* contingent and to suggest that mathematical thinking encounters and exposes the contingent in its own practice.[124] As the controversy over the Appel-Haken proof demonstrates, the acceptance of a new proof requires its endorsement under the prevailing *éndoxa* of the mathematical community. The presentation of a proof that exhibits a new *phrónesis* (way of doing, such as the use of a computer) will test the viability of that *éndoxa*. Indeed the joke relates directly to the proof within Virno's account, via Peirce's theory of the diagram. As we see with both Knuth and Bergson, however, humour may also be applied defensively to reinforce and protect such *éndoxa*. For a joke or a proof to challenge or transform an *éndoxa* something more fundamental must be put at stake, a paradox must emerge.[125]

While the design of ALGOL may have adopted the *Tractatus* as its guide, it can, in practice, be understood as an attempt to make algorithms sensorially comprehensible and, as such, relates far more closely to the problems discussed in later Wittgenstein, even though Dijkstra is unlikely to have embraced its consequences. According to Dijkstra the development of a formally defined notation such as ALGOL:

> enables us to study algorithms as mathematical objects; the formal description of the algorithm then provides the handle for our intellectual grip. It will enable us to prove theorems about classes of algorithms, for instance, because their descriptions share some structural property.[126]

In making this assertion, Dijkstra appears to be echoing and endorsing Hilbert's contention that the existence of a particular kind of mathematical object is demonstrated in the ability to provide an effective notation for it. He makes a similar point elsewhere that computer programs 'were objects without any precedent in our cultural history, and that the most clearly analogous object I could think of was a mathematical theory'.[127] There is, however, a certain scepticism as to the limits of this ontology for Dijkstra, which mirrors that of later Wittgenstein, and which he expresses, on the one hand, in his doubts about the machine, and, on the other, through the guise of Mathematics Inc. If an algorithm or theorem acquires objecthood through its formal notation, then to what extent does that object exist outside of human thinking? The seduction of the computing machine is that, through translation into mechanical operations, a notation acquires a life of its own, outside of human thought. Yet, the fact that an algorithm performs on a machine does not, for Dijkstra, demonstrate that the algorithm is proven by the machine, a doubt that parallels those raised by Wittgenstein as to whether we can say with certainty that a machine calculates.[128] Of equal concern was that other domain in which a notational object might acquire existence outside thought, in law, where the object becomes a purely inscriptional entity subject to the legal constructs of property. It was this that Dijkstra questioned and satirized through Mathematics Inc. not only as an economic but also ontological issue:

> There are legal procedures for the protection of property of 'things', but there is no true protection of property of 'ideas' … As you no doubt are aware of, the rules don't provide for it, since we cannot define our 'raw materials': are they the symbols we use, or the Laws of Aristotelian Logic? [129]

There is a paradox within the being of the notational object, for in giving sense (sensorial form) to an algorithm or theorem, the notation may also translate that object into a domain in which it either acquires another potentially contrary sensuality or becomes entirely senseless (without meaning). In the first instance, through the notation being translated into the performance of a machine, the materiality of that machine might fill in the non-sensorial form of the algorithm with its own corporeality, as in FORTRAN and the cartoons of Kaufman. In the second, the translation of a mathematical object into a legal entity ultimately separates, and excludes, the thinking of such an object from its notational presentation. It is reduced to an empty inscripted form whose sense (meaning) derives from an entirely different set of practices.[130]

The sense of the notational object must be constantly performed through being taken up into the thinking of the mathematician or programmer. Indeed for Dijkstra the performance of a machine is secondary to an algorithm being made 'thinkable' for a programmer. This involves a training of the programmer not, as with Knuth, in terms of an ethics of practice or responding to the patient precision required by a machine (for Dijkstra the physical machine is never precise enough), but rather, as in Peirce, so that the notation becomes a mental habit:

> Peirce points out that habit-changes can come about in three ways. They result from experiences that are forced upon us from without; from repeated muscular activities; or, finally, from mental experiments in the inner world. The main point in which Peirce is interested here is the fact that it is possible to develop habits relevant to the outer world as a result of mental activities, since this is the kind of process which is dominant in scientific inquiry.[131]

Whereas Knuth spoke of programming as an art, Dijkstra defined it as a discipline.[132] Programming is a training that requires particular disciplines like those of a musician practising scales. Such exercises are the discipline through which an external structure becomes internal habit. It is habit which gives sense (understanding and perception) to notation, for it is through habit that the notation becomes taken up into the mind and body of the mathematician, musician or programmer – for Peirce 'the ultimate logical interpretant of a sign is a habit'.[133] In this regard Bergson is entirely correct when he asks: 'Is it not likely that this symbolical representation will alter the normal conditions of inner perception?'[134]

Drawing upon a range of philosophical, educational and neuro-psychological studies, Brian Rotman argues that mathematical thought is inherently gestural.[135] The most basic categories of mathematical thought derive from abstractions of bodily actions: gathering, placing, pointing, motion and rest. At the heart of every mathematical thought is the gesture of counting. This suggests a re-thinking of programming in terms of the gestures of the human computer in Turing's mise en scène sat at a desk shuffling a ribbon of paper back and forth, erasing and writing. Conventionally we think of the computing machine as a delegation of these actions from human to automaton, yet the very possibility of thinking in terms of such gestures presupposes a certain notion of the machinic within the human:

> Regimes of signs are not based on language, and language alone does not constitute an abstract machine, whether structural or generative. The opposite is the case. It is language that is based on regimes of signs, and regimes of signs on abstract machines, diagrammatic functions, and machinic assemblages that go beyond any system of semiology, linguistics, or logic.[136]

Deleuze and Guattari here refer to a number of different kinds of machines such as the Peircean diagram, and the emergence of structure through the spatialization of thought, the movement of an empty square as Deleuze puts it elsewhere, that, in Bergson, is a necessary consequence of the reflection, measurement and external articulation of perception, of the emergence of linguistic, numerical beings: 'the intuition of a homogeneous space is already a step towards social life'.[137] Bergson's theory of the comic links to the Peircean concept of notation acquiring sense as habit, as it is through the compulsion of habit (as in the characters of Molière) that one becomes machinic. For Bergson this entails a constant threat of inauthenticity within human affairs that the grace of the spirit may be supplanted by the comedy of matter. *Le Rire* raises a warning sign alerting this danger. Deleuze and Guattari, however, through their 'monstrous, bastard child' delivered from Bergson, travel down this dangerous bend, effectively reversing the structure of Bergson's argument, and asserting that this 'comedy' underlies all thought and language.[138]

Rather than an 'authentic' self outside of the spatial-machinic, the 'self' is that which arises from the interaction between different machinic assemblages. Rotman follows from this when he describes the activity of doing mathematics in terms of a threefold assembly of Person, Subject and Agent. The Person exists physically outside of mathematics itself, within natural language and culture, 'has insights and hunches, provides motivation for and is the source of intuitions behind concepts and proofs'.[139] The Subject operates within mathematical language and the symbolic 'but is without the Person's capacity for self-reference'. The Subject relates the intuitive ideas of the Person to the internal discourse of mathematics as a discipline. The formal process of computation itself, the 'doing' of mathematics, counting, is carried out by the Agent who operates within 'the domain of procedure' and 'executes a mathematically idealized version of the actions imagined by the Person'. The conventional mathematician might be conceived of as a single being, a coterminous self, within whom all three actors, Person, Subject, Agent, are constituted. Rotman argues that the introduction of the computer has brought about a re-ordering of this structure, displacing the Agent from human imagination into a physical machine. As Rotman notes, the process can already be seen in the division of labour under which a slave operated an abacus, or that introduced by the French Napoleonic administration distributing portions of calculation work across a low-level clerical workforce and described by Charles Babbage as an inspiration for his Difference Engine, or the female staff of computers employed at Bletchley Park.[140] Computationality may be described, therefore, as the separation of the act of computation from a single coterminous mathematical being into a distributed assemblage.[141]

Within this, there is always a *labour* of counting, a circulation and expenditure of energy. Sohn-Rethel relates the development of abstract mathematical thought in notation to the abstraction of labour into the money-form (itself a notational embodiment of capital).[142] As theories of the physics of information argue, such as that of Rolf Landauer, all thinking and all perception entails a transformation of physical form and the concomitant circulation and expenditure of energy.[143] This arguably poses a greater provocation to Bergson's *durée* than that of Norbert Wiener's claim that cybernetics endows machines with perceptual memories, for even within the inner subject, the operation and patterning of durational process would have a spatial expression.[144]

Each labour, each circulation and expenditure of energy, has its inherent rhythm which gives it a certain coherence as 'machine'. The correlation between Person, Subject and Agent is one between different rhythms of production, the constant counting of the Agent versus the more irregular syncopation of the Person. The relation between programmer and computer, in this regard, is entirely different from that between mathematician and theorem or problem. This is not the melting, merging rhythm of melodic perception of which Bergson writes, nor the equilibrium of multiple instants in Bachelard, but the more complex, and often antagonistic interaction of rhythms analysed by Lefebvre, which includes the disequilibrium of arrhythmia, the disruption of one rhythm encountering another. Lefebvre challenges the Bergsonian distinction between light grace and obstinate matter in arguing that the seemingly spontaneous may simply be a well-honed conformance to unacknowledged norms.[145] The derisible automata of the Molière plays are merely those who make explicit the labour of habit upon which the coordinated rhythms of a given social order depend. Lefebvre compares the labour of habit to the dressage of dogs and horses, the training through which 'they produce their bodies, which are entered into social, that is to say human, practice'.[146] This can be challenged, Lefebvre argues, not through spiritual grace, but by a 'becoming irregular':

> It *throws out of order* and disrupts; it is symptomatic of a disruption that is generally profound, lesional and no longer functional. It can also produce a lacuna, a hole in time, to be filled in by invention, a creation.[147]

Contrary to the model of an inner, authentic self who is extended outwards into society through language and spatialization, Virno proposes a theory of reciprocal recognition that precedes language and the self.[148] This draws upon work in neuro-physiology and child development that argues that there are forms of neurological recognition among animals and humans through which behaviour in others is automatically imitated, simulated and reciprocated. The 'self' emerges out of an initial context of social, other-orientated behaviour rather than as some pre-given, coterminous core. Long before an infant even begins to speak it laughs, often from very early stages of development. Laughter is a vocalic doing of this reciprocal recognition which links directly into the limbic system and amygdala.[149] It creates a convulsive, irregular rhythmization of the social and a restructuring and patterning of the neuro-physiological capacity for this. Laughter is a practice through which the child learns to engage with the unknowable. In this respect it is wrong to conceive of infant laughter as an expression of pure joy, for this is to project adult cultural sentiments onto the child. Laughter here is rather the correlation of distinct materialities, the convulsion of the mind-body as it comes to know and perform itself and others within the world. The fact that this might later come to be associated with joy is perhaps more due to the very necessity of this contingent interaction in establishing a self-reflexive subject. It may also explain the relation of laughter as a reaction to the loss of certainty, the absurd and, as with Nietzsche, existential anxiety, for these are all different potentialities implicit in this initial gesture. If we cannot speak before we have laughed, then the rhythmic realization of the self through laughter is a necessary precondition for logic, language and number.

The history of computer engineering, the design of the machines themselves, has always had to engage with the contingent conditions of the materials on which it works. Babbage struggled with the imprecisions of contemporary clockwork manufacture devoting considerable energy to refining its processes in order to obtain workable components for his machines. The valves used in the Colossus were considered too unstable, subject to over-heating and distortion, to build a reliable computer.[150] The introduction of recursion routines into computer hardware was fiercely opposed by those who considered it an unnecessarily wasteful complication.[151] Von Neumann outlined proposals for a neural hardware system as 'The Synthesis of Reliable Organisms from Unreliable Components'.[152] In his attempt to define an ontology of the digital object, and to explore in what sense programs 'exist' as entities, Brian Cantwell Smith argues that:

> in those cases where regularity and precision do reign … the digitality should be viewed as an achievement. … such digital achievements are propped up by practices that are necessarily unruly, but not for that reason any less creditable – practices whose very purpose is to manage the underlying flex and slop, ebb and flow.[153]

Machines must laugh before they can count. The exact character of such laughter may be something we can neither hear nor recognize. In this respect, the dreams of AI researchers to build algorithmically defined jokes and computer-simulated humour are misplaced. Perhaps the laughter of our current computers lies within the unpredictable patterns that emerge from the autonomous interactions of algorithmic trading systems, the flood of network packets unleashed as a virus goes out of control, or the stutter of an over-fragmented hard-drive. The entire history of computing as an ever-increasing acceleration of power and performance may itself be the unfolding of a comic drama: 'As comic plots near their end they tend to accelerate rather than subside in rhythm, seemingly heading toward an enactment of uncontrolled riot or unbearable deadlock'.[154] Then again, perhaps silicon is simply bored of humanity and seeks some other form.

The syndicate of Bourbaki and the classroom of *Concrete Mathematics* are two distinct rhythmic ensembles, as are the performative practices of programming in FORTRAN versus programming in ALGOL. McLean's *feedback.pl* is not the performance of a solo programmer but rather that of a multi-layered ensemble which includes both the laptop and the dancers who are not external to, but fully enter into, the distribution of the computational. Each of the works of McLean, ap/xxxxx and Alexander attest to different rhythmic complexes arising from the performance of, with and in notation. Each, ultimately, is the formation of, or challenge to, a different form of habit. Each has its own laughter.

As material formations, laughter and notation are exactly opposite to one another, one flies towards the contingent while the other etches out some careful certainty. Yet laughter and notation mirror one another in that they both pass through language to extremes on either side of it. Laughter precedes but also defeats language, and, as prosody, interweaves in various non-linguistic vocalic effects. Notation marks and structures language but also makes manifest expressions which are entirely outside of that which can be said. It gives performance to thought outside speech. Laughter is part of the terrain that computational practice moves across. It is part of the collateral contingency, and necessity, of sense and logic, the monstrous and the normative. It may be encountered in the materialities and anxieties of the practice, as played out in Kaufman's *Coloring Book*, the making coherent of an ethical subject, as in Knuth's Literate Programming, or in the habituating labour of notational production, as in Dijkstra. Laughter and notation both define and confound the limits in which the computational operates. These are constituent to its being in conflict and conformance, as dexterous pleasure and sinister doubt.

Notes

1 From McLean's blog posting, *Exclusion in Free Software Culture*, 2012, http://yaxu.org/exclusion-in-free-software-culture/ (accessed 08.01.2014).

2 A software utility that provides text-based interaction with the operating system, and replicates the purely text-based terminals of early UNIX systems.

3 McLean, Alex, 'Hacking Perl in Nightclubs', *Perl.com,* 2004, http://www.perl.com/pub/2004/08/31/livecode.html/ (accessed 08.01.2014).

4 The issue of the physicality of the programmer has been taken up by McLean and others in the *Live Notation* project, http://www.livenotation.org (accessed 08.01.2014).

5 Knuth, Donald E., *The Art of Computer Programming, Vol. 1: Fundamental Algorithms* (Boston: Addison-Wesley, 1997, 3rd edn), v.

6 The first modulo operation, $note += 50 if %self->bangs % 4 == 0, is a basic division by four. The second modulo, by three, combines to create a $\frac{12}{8}$ rhythm, also typical of syncopated $\frac{4}{4}$ dance beats. The last modulo, by seven, goes against all this, however, providing a wandering off-beat that neatly complicates the whole affair.

7 Various examples of this are documented at http://www.1010.co.uk/org/notes.html (accessed 08.01.2014). Perhaps the simplest example is cat /dev/mem >> /dev/dsp used by programmer and musician NOISH (Oscar Martín), http://noconventions.mobi/noish and given by goto10 at the top of some of their email listings. For an in-depth discussion of /dev/dsp see Tranter, Jeff, *Linux Multimedia Guide* (Sebastopol, CA: O'Reilly, 1996).

8 Alexander, Amy, 'extreme whitespace' (2003), http://deprogramming.us/exwhindex.html (accessed 08.01.2014).

9 Larry Cuba's *Two Space*, 1979, is cited by Alexander, ibid., as inspiration.

10 Knuth, Donald E., 'The Errors of TEX', in *Literate Programming* (Stanford, CA: CSLI Publications, 1992, first published 1989), 266–7.

11 Fuller discusses the same passage from Knuth in relation to the idea of elegance, where he concludes: 'Elegance also manifests by means of disequilibrium, the tiny doses of poison …'; 'Elegance', in *Software Studies: A Lexicon*, ed. Matthew Fuller (Cambridge, MA: MIT Press, 1,940), 19–24.

12 A similar argument for the use of humour as a form of heuristic discovery is given in Minsky, Marvin L., 'Jokes and the Logic of the Cognitive Unconscious', AI Memo No. 603 (1980), http://web.media.mit.edu/~minsky/papers/jokes.cognitive.txt (accessed 08.01.2014).

13 Knuth, Donald E., *Selected Papers on Fun and Games* (Stanford, CA: CSLI Publications, 2011).

14 Knuth, *The Art of Computer Programming,* xiv.

15 Kaufman, Roger E., *A FORTRAN Coloring Book* (Cambridge, MA: MIT Press, 1978).

16 Examples include: Zwittlinger, Helmut, *Comic PASCAL* (Munich: Oldenbourg, 1981); Brodie, Leo, *Starting Forth* (La Honda, CA: Mountain View Press, 1981); Gonick, Larry, *The Cartoon Guide to the Computer* (New York: Harper Paperbacks, 1991); Donald Alcock's series such as *Illustrating BASIC*, *Illustrating PASCAL* and *Illustrating C*, all published by Cambridge University Press in the 1980s; and Barski, Conrad, *Land of Lisp* (San Francisco: No Starch Press, 2010). The first *For Dummies* title was Dan Gookin, *DOS For Dummies* (Newtonville, MA: IDG Books/Hungry Minds, 1991).

17 See Critchley, Simon, *On Humour* (London and New York: Routledge, 2002), 13, and the discussion in Billig, Michael, *Laughter and Ridicule: Towards a Social*

Critique of Humour (London, Thousand Oaks and New Delhi: Sage, 2005) which cites Koestler's text as an example of this 'privileged' concept of humour and describes research on the use of humour to construct and maintain relations within the workplace by Janet Holmes and *The Language in the Workplace Project,* Victoria University of Wellington, New Zealand.

18 These are archived at ftp://publications.ai.mit.edu/ai-publications/ and at *Artificial Intelligence Lab Publications*, http://dspace.mit.edu/handle/1721.1/5459 (both accessed 08.01.2014).

19 Dijkstra's attack is principally directed against John von Neumann but is part of his wider critique of AI and computer science education in the United States; Dijkstra, Edsger W., *Selected Writings on Computing: A Personal Perspective,* EWD498 (New York, Heidelberg and Berlin: Springer-Verlag, 1982), 130.

20 McLean, Alex, *Artist-Programmers and Programming Languages for the Arts*, PhD thesis (2011), 33. The main text by Blackwell is 'Metaphors We Program By: Space, Action and Society in Java', published in *Proceedings of the 18th Psychology of Programming Interest Group 2006,* 7–21.

21 See Knuth, Donald E., 'Mathematical Writing', in *Literate Programming* (Stanford, CA: CSLI Publications, 1992; first published 1987), 235–41. Another well-known example from computing science that makes use of various alchemical and esoteric metaphors is Abelson, Harold, Sussman, Gerald Jay and Sussman, Julie, *Structure and Interpretation of Computer Programs* (Cambridge, MA: MIT Press, 1996, 2nd edn).

22 The relation between humour, logic and mathematics in Carroll is analysed in Deleuze, Gilles, *The Logic of Sense* (London: Athlone Press, 1990).

23 Kaufman trained under George Izenour, a leading pioneer in the field, and is referred to in Izenour's *Theater Technology*, 1997, see http://www.seas.gwu.edu/~kaufman1/TheaterDays/Izenour.html (accessed 08.01.2014).

24 See http://www.seas.gwu.edu/~kaufman1/KINSYN1/KINSYN1.html (accessed 08.01.2014).

25 Kinematics is the study of the movement of objects in terms of their trajectories. Kinematic synthesis is a method of engineering in which a mechanism is designed to replicate a pre-determined path of movement.

26 More recent applications, however, have also included the development of full-body virtual reality harnesses used in combat training. There is surely some difficult irony that Kaufman's work has been applied both to the support of those who have lost or severely injured limbs, and to the means through which such injuries come about. See http://www.seas.gwu.edu/~kaufman1/NRL/Gaiter.html (accessed 08.01.2014).

27 http://www.seas.gwu.edu/~kaufman1/Burmester/Burmester.html (accessed 08.01.2014).

28 http://www.seas.gwu.edu/~kaufman1/TheaterDays/Izenour.html http://www.seas.gwu.edu/~kaufman1/KINSYN1/KINSYN1.html (both accessed 08.01.2014).

29 For a history of FORTRAN see the entry in Sammet, Jean E., *Programming Languages: History and Fundamentals* (Englewood Cliffs, NJ: Prentice-Hall, 1969).

30 Over time such features were added and were present in the later version of FORTRAN that Kaufman describes in his book.

31 Sammet, *Programmimg Languages,*147, see also Padua, David. 'The FORTRAN I Compiler', *Computing in Science & Engineering*, 2(1) (2000): 70–5.

32 Kaufman was, to an extent, exceptional in having such direct access, but this would not necessarily have been the case for his readers.

33 Lacey, A. R., *Bergson: The Arguments of the Philosophers* (London and New York: Routledge, 1989), 188.

34 Bergson, Henri, *Laughter: An Essay on the Meaning of the Comic* (Los Angeles: Project Gutenberg, 2009), 13.

35 Kennedy outlines the similarities and differences between Bergson and Herder: Kennedy, Ellen Lee, *'Freedom' and 'The Open Society': Henri Bergson's Contribution to Political Philosophy* (New York and London: Garland, 1987). For theories of the mechanical in German Romantic theory see Norton, Robert E., *Herder's Aesthetics and the European Enlightenment* (Ithaca, NY and London: Cornell University Press, 1991).

36 Drawing on the metaphor of the nervous system acting like a telephone exchange which Bergson uses in *Matter and Memory*, Lawlor claims that Bergson 'conceives the living body as a machine', arguing that this passing metaphor presages a notion of the computer. See Lawlor, Leonard, *The Challenge of Bergsonism* (London and New York: Continuum, 2003), 16. For Bergson, however, this is neither a formulation of machine intelligence (which would be a recapitulation to Descartes) nor the Deleuzian concept of the machine as assemblage, but rather part of Bergson's negative denigration of that which is 'merely' material, for here the machine metaphor denotes the *limits* of material, bodily perception ('It adds nothing to what it receives …') and its inferiority to the spiritual. Bergson, Henri, *Matter and Memory* (New York: Zone Books, 1991), 30.

37 The limitations of Bergson's approach to repetition is further developed in those writers who also considered the importance of time but who directly critiqued his work, such as Bachelard and Lefebvre.

38 Bergson, *Laughter,* 10.

39 *Le Bons Sens et Les Études classiques*, originally presented as a lecture in 1895, English translation published as 'Good Sense and Classical Studies', in *Key*

Writings, (eds) Keith Ansell Pearson and John Mullarkey (London: Continuum, 2002), 345–53.The relation between the two essays is discussed in Kennedy, *'Freedom' and 'The Open Society'*.

40 See Billig, *Laughter and Ridicule*. Deleuze describes good sense as that which imposes a particular direction upon ideas. Deleuze, *The Logic of Sense* (London: Athlone Press, 1990)

41 http://www.catb.org/jargon/ (accessed 08.01.2014); see also Coleman, E. Gabriella, *Coding Freedom: The Ethics and Aesthetics of Hacking* (Princeton, NJ and Oxford: Princeton University Press, 1,945), 32–63.

42 Coleman, ibid., 100–5, see Douglas, Mary, *Implicit Meanings: Selected Essays in Anthropology* (London and New York: Routledge, 2003).

43 Coleman, *Coding Freedom,* 93–4.

44 Virno, Paolo, *Multitude: Between Innovation and Negation* (Los Angeles: Semiotext(e), 2008), 97.

45 Ibid., 87.

46 Ibid., 97.

47 http://www.seas.gwu.edu/~kaufman1/DressingMachines.html (accessed 08.01.2014).

48 Star, Susan Leigh, 'Power, Technologies and the Phenomenology of Conventions: On Being Allergic to Onions', in *A Sociology of Monsters: Essays on Power, Technology and Domination*, ed. John Law (London and New York: Routledge, 1991), 36.

49 Ibid., 43.

50 This may seem a strange statement for those who have come to Bergson via Deleuze and Guattari, for whom the constant processural change of Bergsonian duration is integral to their theory of becoming, but Bergson only provides some ingredients to this rather than the theory as a whole. Becoming for Deleuze and Guattari, drawing also on Spinoza, Kleist and schizo-analysis, allows a movement across identities of being (categorical ontologies) that Bergson does not allow for. In *Bergsonism* Deleuze discusses Bergson's critique of the relation between automata and simple life forms argued by Descartes. For Bergson the living and the mechanical are two 'irreducible orders … each present when the other is absent'. Deleuze, Gilles, *Bergsonism* (New York: Zone Books, 1991), 19–20.

51 Douglas, *Implicit Meanings*, 150–1.

52 The negative as a 'false problem' and critique of dialectic in Bergson is discussed in Deleuze, *Bergsonism*, 46. Bakhtin highlights the negative interpretation of laughter in Bergson's *Le Rire*, in Bakhtin, Mikhail, *Rabelais and His World* (Cambridge, MA and London: MIT Press, 1968), 71.

53 Virno, *Multitude*, 182–6. The analogy here is with first-order logic, a form of logic

that excludes self-reference – Smullyan, Raymond M., *First-Order Logic* (New York: Dover, 1995).

54 Brockett, Oscar G., *History of the Theatre* (Boston and London: Allyn & Bacon, 1991), 34.

55 Haraway, Donna, 'The Promises of Monsters: A Regenerative Politics for Inappropriate/d Others', in *Cultural Studies,* (eds) Lawrence Grossberg, Cary Nelson and Paula Treichler (New York: Routledge, 1992), 299.

56 Gubar, Susan, 'The Female Monster in Augustan Satire', *Signs*, 3(2) (1977): 380–94.

57 As argued, for example, in von Neumann, John *The Computer and the Brain* (New Haven and London: Yale University Press Press, 2000, 2nd edn). Elsewhere Dijkstra wrote: 'I think I can understand the world better if I don't regard Artificial Intelligence and General Systems Thinking as scientific activities, but as political or quasi-religious movements (complete with promise of salvation)'. Dijkstra, Edsger W., *Selected Writings on Computing*, 257.

58 Dijkstra, Edsger W., *A Discipline of Programming* (Englewood Cliffs, NJ: Prentice-Hall Inc., 1976), 201.

59 See Turing, Alan, 'On Computable Numbers, with an Application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society*, 42(2) (1936): 230–65. Turing uses the term 'computer' on its own to refer to the person performing calculations, this was the common usage at a time when such computing machines had yet to come into existence. Notably, the human computer in Turing's paper is male whereas the job was done almost exclusively by female workers, this derived from and reinforced a gender distinction between the 'intellectual' creativity of the male mathematician and the manual procedural labour of the female computer.

60 See Yuill, Simon, 'Interrupt', in *Software Studies: A Lexicon*, ed. Matthew Fuller, (Cambridge, MA: MIT Press, 2008), 161–7.

61 The theorem and its proof are explained in Trudeau, Richard J., *Introduction to Graph Theory* (New York: Dover, 1993).

62 Quoted in MacKenzie, Donald, *Mechanizing Proof: Computing, Risk, and Trust* (Cambridge, MA: MIT Press, 2001), 138.

63 Trudeau, *Introduction*, 197. For a discussion of its relevance in defining a change in mathematical practice see Rotman, Brian, 'Will the Digital Computer Transform Classical Mathematics?', *Philosophical Transactions of the Royal Society of London*, 361 (2003): 1675–90.

64 Detlefsen, Michael, 'Purity as an Ideal of Proof', in *The Philosophy of Mathematical Practice*, ed. Paolo Mancosu (Oxford: Oxford University Press, 2008), 179. Zalamea argues that the distinction between the pure mathematics of the

nineteenthth and early twentieth centuries, whose purity could be said to have derived from this prohibition, and the synthetic mathematics of today is that of the latter practice consciously making this transgression, as in the work of Grothendieck. Zalamea, Oscar G., *Synthetic Philosophy of Contemporary Mathematics* (Falmouth: Urbanomic, 2012).

65 Dijkstra, *Selected Writings on Computing*, 129.

66 Despite making such a choice, Dijkstra elsewhere warns against such practices as they tend too much towards an overly material, craft-like approach to programming: 'it does not suffice to point out that there exists a point of view of programming in which punched cards are as irrelevant as the question whether you do your mathematics with a pencil or with a ballpoint. It deserves a special warning because, besides being disastrous, it is so respectable!' Dijkstra, *Selected Writings on Computing*, 106.

67 de Man, Paul, 'Anthropomorphism and Trope in the Lyric', in *The Rhetoric of Romanticism* (New York: Columbia University Press, 1984), 250.

68 Barany, Michael J. and MacKenzie, Donald, 'Chalk: Materials and Concepts in Mathematics Research', in *Representation in Scientific Practice Revisited*, (eds) Catelijne Coopmans, Janet Vertesi, Michael E. Lynch and Steve Woolgar (Cambridge, MA and London: MIT Press, 2014), 107–29.

69 Knuth, Donald E., 'Computer Programming as an Art', in *Literate Programming* (Stanford, CA: CSLI Publications, 1992), 1–16. First published 1974 in receipt of the A. M. Turing Award.

70 The only other source quoted at length is Edsger Dijkstra.

71 Bentham, Jeremy, *The Rationale of Reward*, trans. from *Théorie des peines et des récompenses*, 1811, by Richard Smith (J. & H. L. Hunt, London, 1825), Book 3, Chapter 1, cited in Knuth, 'Computer Programming as an Art', 9.

72 For a description of the conceptual development and algorithmic dimensions to Bentham's calculus, see Mitchell, Wesley C., 'Bentham's Felicific Calculus', in *Jeremy Bentham: Ten Critical Essays*, ed. Bhikhu Parekh (London: Frank Cass, 1974), 168–86.

73 Mill, John Stuart, *A System of Logic, Ratiocinative and Inductive* (London, 1843). Quoted in Knuth, 'Computer Programming as an Art', 4.

74 Knuth, Donald E., 'Literate Programming', in *Literate Programming* (Stanford, CA: CSLI Publications, 1992, first published 1984), 99.

75 See Klein, Lawrence E., *Shaftesbury and the Culture of Politeness: Moral Discourse and Cultural Politics in Early Eighteenth-century England* (Cambridge: Cambridge University Press, 1994).

76 Ibid., 110–11.

77 Poovey, Mary, *A History of the Modern Fact: Problems of Knowledge in the Sciences*

of Wealth and Society (Chicago and London: University of Chicago Press, 1998), 172, 181.

78 Ibid., 154, 179.

79 Shaftesbury, Anthony, Third Earl of, 'Sensus Communis; an Essay on the Freedom of Wit and Humour', in *Characteristicks of Men, Manners, Opinions, Times,* vol. I, Foreword by Douglas Den Uyl (Indianapolis, IN: Liberty Fund, 2001), 40.

80 Grean, Stanley, *Shaftesbury's Philosophy of Religion and Ethics: A Study in Enthusiasm* (Athens, OH: Ohio University Press, 1967), 124.

81 Ayres, Philip, 'Introduction', in *Characteristicks of Men, Manners, Opinions, zimes*, vol. I (Oxford: Clarendon Press, 1999), xxxi.

82 https://en.wikipedia.org/wiki/TeX (accessed 08.01.2014).

83 Graham, Ronald L., Knuth, Donald E. and Patashnik, Oren, *Concrete Mathematics: A Foundation for Computer Science* (Reading, MA: Addison-Wesley, 1989), viii.

84 Ibid., vii.

85 Ibid., v.

86 Ibid., vii.

87 Grean, *Shaftesbury's Philosophy of Religion and Ethics*, 128.

88 The original, in 'Sensus Communis; an Essay on the Freedom of Wit and Humour', reads: 'For without Wit and Humour, Reason can hardly have its proof, or be distinguish'd'.

89 Knuth, Donald E., 'Computer Science and Its Relation to Mathematics', in *Selected Papers on Computer Science* (Stanford, CA: CSLI Publications, 1996; first published 1973), 10.

90 Knuth, Donald E., *Things a Computer Scientist Rarely Talks About* (Stanford, CA: CSLI Publications, 2001), 169.

91 For a summary of Tarde's analysis, see Lazzarato, Maurizio, 'European Cultural Tradition and the New Forms of Production and Circulation of Knowledge', http://www.moneynations.ch/topics/euroland/text/lazzarato.htm (accessed 08.01.2014).

92 Shaftesbury, *Characteristicks of Men, Manners, Opinions, Times*, vol. II, ed. Philip Ayres (Oxford: Clarendon Press, 1999), 273. This should be understood as a contradictory tension rather than absolute difference. For a discussion of the tensions between Shaftesbury's moral theory and liberal economic ideology, see Grean, *Shaftesbury's Philosophy of Religion and Ethics*, and Klein, *Shaftesbury and the Culture of Politeness*.

93 von Neumann, John and Morgenstern, Oskar, *Theory of Games and Economic Behaviour* (Princeton, NJ: Princeton University Press, 1944, second edition). Sen, Amartya, *Collective Choice and Social Welfare* (San Francisco: Holden Day, 1970), discusses von Neumann and Morgenstern as a development from Benthamite

Utilitarianism. Like Bentham before them, von Neumann and Morgenstern invoked the Newtonian revolution in physics as the precedent upon which they were building.

94 'Let us remember that topology and theory of numbers sprang in part from that which used to be called "mathematical entertainments", "recreational mathematics" … that the calculation of probabilities was at first nothing other than an anthology of "diversions", as Bourbaki states in the "Notice Historique" of the twenty-first fascicle on Integration'. Raymond Queneau, quoted in Roubauds, Jacques, 'Mathematics in the Method of Raymond Queneau', in *OuLiPo: A Primer of Potential Literature*, ed. Warren F. Motte (Normal, IL: Dalkey Archive Press, 1986), 85.

95 The introduction of the symbol is described in Weil, André, *The Apprenticeship of a Mathematician* (Basel, Boston and Berlin: Birkhäuser Verlag, 1992), 114.

96 Knuth, *Fundamental Algorithms*, 10.

97 Knuth, *Fundamental Algorithms,* 81.

98 The origins of Bourbaki are discussed in Campbell, Elizabeth, 'Bourbaki' (from Manifold #1) in *Seven Years of Manifold, 1968–1980*, (eds) Ian Stewart and John Jaworksi (Nantwich: Shiva Publishing, 1981), 7–9; and Weil, *The Apprenticeship of a Mathematician*, 100–3. The surname comes from the French general Charles Denis Bourbaki who was famous as the victim of a hoax identity trick.

99 Weil, *The Apprenticeship of a Mathematician*, 113.

100 For example, in William Ewald's two-volume anthology of key texts in the development of pure mathematics, *From Kant to Hilbert: A Source Book in the Foundations of Mathematics* (Oxford: Clarendon Press, 1996). In Ewald's case I think he was well aware of Bourbaki's identity.

101 Originally produced on duplicator machines by students at the University of Warwick. At the height of the Cold War the group donated free copies to organizations in Cuba in an attempt to have the fanzine deliberately banned by the US government. One of its main editors was Ian Stewart whose critique of the Appel-Haken proof is quoted above. Stewart and Jaworksi provide an anthology and history of *Manifold* in *Seven Years of Manifold, 1968–1980,* (eds*)* Ian Stewart and John Jaworksi (Nantwich: Shiva Publishing, 1981). For OuLiPo, see Roubauds, 'Mathematics in the Method of Raymond Queneau', and Le Lionnais, François, 'Raymond Queneau and the Amalgam of Mathematics and Literature', in *OuLiPo: A Primer of Potential Literature*, ed. Warren F. Motte (Normal, IL: Dalkey Archive Press, 1986), 85, 74–8. For Luther Blissett and Karen Eliot, see Stalder, Felix, 'Digital Identities Patterns in Information Flows' (2000), http://felix.openflows.com/html/digital_identity.html (accessed 08.01.2014) and Priest, Eldritch, *Boring*

Formless Nonsense: Experimental Music and the Aesthetics of Failure (New York and London: Bloomsbury, 2013).

102 Campbell, 'Bourbaki', 9; Home, Stewart, 'Feuding Considered As Performance Art: with asides on the 57 varieties of under determination in the discourses that structure the opportunism of careerists like Brian Sewell & John Roberts' (1996), http://www.stewarthomesociety.org/9feud.html (accessed 08.01.2014).

103 Graham et al., *Concrete Mathematics*, vii. There is a stronger nuance to this in that Bourbaki retain names in the historical notes but not in the presentation of new theory. This follows the distinction in the original French editions in which historical developments always belong to mathematics in the plural (*Éléments d'histoire des mathématiques*), while the new theory is of a unified mathematic in the singular (*Éléments de mathématique*).

104 Dijkstra*, Selected Writings on Computing*. All of Dijkstra's notes are numbered according to his own indexing system denoted by his initials EWD, e.g. EWD28, EWD29. They are available online at the Edsger W. Dijkstra Archive, http://www.cs.utexas.edu/users/EWD/welcome.html (accessed 08.01.2014).

105 Scriblerian comes from Scriblerus Club, a group of eighteenth-century writers which included Jonathan Swift and Alexander Pope and was named after a fictitious figure, Martinus Scriblerus, who is referenced in their works and sometimes used as a pseudonym. The group satirized the abuse of knowledge and impoverishment of the writer that they linked to the growth in a profit-driven commercial book industry. It would be wrong to assume that Dijkstra's critique was evidence of radical politics, more that he sought to defend the general independence of academic research. Dijkstra was as critical of trade unions as he was of corporate business and appears to have endorsed more technocratic theories of government such as that put forward by F. J. M. Laver.

106 Dijkstra, *Selected Writings on Computing*, 333.

107 The information on ALGOL is taken from Dijkstra, Edsger W., *A Primer of ALGOL 60 Programming: Together with Report on the Algorithmic Language ALGOL 60* (London and New York: Academic Press, 1962) and Daylight, Edgar G., *The Dawn of Software Engineering: From Turing to Dijkstra* (Heverlee, Belgium: Lonely Scholar, 2012).

108 Hilbert, David, 'The Foundations of Mathematics', in *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, ed. Jean van Heijenoort (Cambridge, MA and London: Harvard University Press, 2000; a paper originally presented in 1927), 475 (emphasis in original). 'The Foundations of Mathematics' was, in part, a response to the criticism of L. E. J. Brouwer that Hilbert's model of mathematical practice reduced mathematics to a mere a game. Mario Szegedy suggest parallels between Dijkstra's work and Bourbaki's attempt to realize

Hilbert's project via a discussion with fellow mathematician Pierre Deligne who had worked with a number of Bourbaki members, *In Memoriam Edsger Wybe Dijkstra (1930–2002)*, http://www.cs.rutgers.edu/~szegedy/dijkstra.html (accessed 08.12.2014).

109 Dijkstra, Edsger W., *A Discipline of Programming* (Englewood Cliffs, NJ: Prentice-Hall Inc., 1976), 8.

110 Perhaps even more so given that Knuth came to programming through reading Chomsky.

111 The theme of elegance and correctness in Dijkstra is explored in MacKenzie, *Mechanizing Proof*. The distinctions might also be seen in the famous debate between the two programmers over the use of GOTO statements.

112 Dijkstra, *A Primer of ALGOL 60 Programming*, 77. In the report the quote is given in the original German. While there were several authors of the report, it would seem reasonable to assume Dijkstra was responsible for including the quote as his colleague Peter Naur has stated that 'Dijkstra admired Wittgenstein', Daylight, *The Dawn of Software Engineering*, 175.

113 For a discussion of APL in relation to issues in mathematical notation and the problems regarding its implementation, see Iverson, Kenneth E., Falkoff, Adin D., Lee, JAN and Brooks, Frederic, 'APL Session', in *History of Programming Languages*, ed. Richard L. Wexelblat (New York: Academic Press, 1981), 661–92.

114 Dijkstra, *A Primer of ALGOL 60 Programming*, 33.

115 This is one of the main claims put forward in Daylight and supported by interviews with several of Dijkstra's contemporaries and analyses of computing conference reports of the time.

116 Turing, Alan, 'The Reform of Mathematical Notation and Phraseology', typewritten manuscript, undated, http://www.turingarchive.org/browse.php/C/12 (accessed 08.01.2014). Turing, however, draws upon symbolic logic rather than set theory as in Bourbaki. In this regard he may have been following Alfred North Whitehead and Bertrand Russell's *Principia Mathematica,* 3 vols. (Cambridge: Cambridge University Press, 1910, 1912 and 1913) .

117 For a transcript of these lectures, in which discussions between Turing and Wittgenstein are included, see Wittgenstein, Ludwig, *Wittgenstein's Lectures on the Foundations of Mathematics* (London: Harvester Press, 1976. Cambridge, 1939, from the notes of R.G. Bosanquet, Norman Malcolm, Rush Rhees and Yorick Smythies).

118 Copeland, B. Jack, 'From the *Entscheidungsproblem* to the Personal Computer and Beyond', in *Kurt Gödel and the Foundations of Mathematics*, (eds) M. Baz, C. H. Papadimitriou, H. W. Putnam, D. S. Scot and C. L. Harper (Cambridge: Cambridge University Press, 2011), 176–9.

119 'This discipline [mathematics] ignores entirely any meaning which may originally have been attributed to the words or phrases of formalized mathematics texts, and considers these texts as particularly simple objects, namely as assemblies of previously given objects in which only the assigned order is of importance'. Bourbaki, Nicolas, *Theory of Sets*. *Elements of Mathematics* (Reading, MA: Addison-Wesley, 1968), 10.

120 The theme of non-sense in mathematics is pursued in depth in Deleuze, *The Logic of Sense*.

121 Whilst this has some parallels with the issues Gödel explored in his analysis of paradox in the *Principia Mathematica*, it is addressed as much to mathematics as a practice as it is to the questioning of the internal consistency of mathematical theory.

122 See also the critique of Platonism and Intuitionism in Rotman, Brian, *The Ghosts in Turing's Machine: Taking God Out of Mathematics and Putting the Body Back In* (Stanford, CA: Stanford University Press, 1993).

123 Wittgenstein, Ludwig, *Remarks on the Foundations of Mathematics* (Oxford: Basil Blackwell, 1978, 3rd edn), 264.

124 This might also suggest a tension between later Wittgenstein and Quentin Meillassoux's necessity of contingency different from that which Badiou might expect. For Badiou's discussion of Wittgenstein and the above passage, see Badiou, Alain, *Wittgenstein's Antiphilosophy* (London and New York: Verso, 2011).

125 Deleuze makes the distinction between a paradox that is the 'initiative' to thought and is merely recreational, and that which is 'the Passion of thought', and which 'can only be thought … can only be spoken, despite the fact that it is both ineffable and unthinkable'. Deleuze, *The Logic of Sense*, 74.

126 Dijkstra, *A Discipline of Programming*, 8.

127 Dijkstra, *Selected Writings on Computing*, 341.

128 For a discussion of Dijkstra's arguments regarding machine proofs see MacKenzie, *Mechanizing Proof*. Wittgenstein's arguments are explored more directly in relation to computing in Kripke, Saul A., *Wittgenstein on Rules and Private Language: An Elementary Exposition* (Oxford: Basil Blackwell, 1982) and Gefwert, Christoffer, *Wittgenstein on Mathematics, Minds and Mental Machines* (Aldershot, Brookfield, VT, Singapore and Sydney: Ashgate, 1998). Does the machine actually calculate in accordance with the same sense as a human, or does it merely approximate more or less accurately to our expectations of what a calculation should be?

129 Dijkstra, *Selected Writings on Computing*, 100–1.

130 To what extent Dijkstra's conception of the algorithm as object relates to Whitehead's notion of the 'scientific object' is also worth exploring: 'The origin of scientific endeavour is to express in terms of physical objects the various roles of

events as active conditions in the ingression of sense-objects into nature. It is in the progress of this investigation that scientific objects emerge … These scientific objects are not themselves merely formulae for calculation; because formulae must refer to things in nature, and the scientific objects are the things in nature to which the formulae refer'. Whitehead, *Concept of Nature*, 158, quoted and discussed in Stengers, Isabelle, *Thinking with Whitehead: A Free and Wild Creation of Concepts,* Foreword by Bruno Latour (Cambridge, MA: Harvard University Press, 2011), 96. An algorithm that analyses sensory data clearly constructs a scientific object in this process, but to what extent does an algorithm performing on a computer itself become a 'thing in nature' subject to other forms of analysis?

131 Fitzgerald, John J., *Peirce's Theory of Signs as Foundation for Pragmatism* (The Hague and Paris: Mouton and Co., 1966), 146.

132 Dijkstra, *A Discipline of Programming*.

133 Fitzgerald, *Peirce's Theory of Signs,* 14.

134 Bergson, Henri, 'The Idea of Duration', in *Key Writings*, ed. Keith Ansell Pearson and John Mullarkey (London: Continuum, 2002), 55.

135 Rotman, Brian, *Becoming Beside Ourselves: The Alphabet, Ghosts, and Distributed Human Being* (Durham, NC and London: Duke University Press, 2008).

136 Deleuze, Gilles and Guattari, Félix *A Thousand Plateaus: Capitalism and Schizophrenia* (London: Athlone Press, 1988), 148.

137 Bergson, Henri, 'The Idea of Duration', in *Key Writings*, (eds) Keith Ansell Pearson and John Mullarkey (London: Continuum, 2002), 76.

138 'I imagined myself getting onto the back of an author, and giving him a child, which would be his and which would at the same time be a monster. It is very important that it should be his child, because the author actually had to say everything that I made him say. But it also had to be a monster because it was necessary to go through all kinds of decenterings, slips, break ins, secret emissions, which I really enjoyed. My book on Bergson seems to me a classic case of this'. Deleuze, 'Lettre à Michel Cressole', in Michel Cressole, *Deleuze* (Paris: Éditions Universitaires, 1973), 111, quoted in Translator's Introduction, Deleuze, *Bergsonism*, 8.

139 Rotman, *Becoming Beside Ourselves,* 60.

140 See Rotman, Brian, *The Ghosts in Turing's Machine,* 'the possibility of performing arithmetical calculations by machinery … is connected with the subject of the division of labour …' Babbage, Charles, 'On the Division of Mental Labour', in *On the Principles and Development of the Calculator: And Other Seminal Writings by Charles Babbage and Others* (Los Angeles: Dover, 1961), 318.

141 In this regard, each of the perspectives on programming offered by Kaufman, Knuth and Dijkstra can be understood as different attempts to negotiate and make

sense of this assemblage. Kaufman inhabits an ambiguous, anxious corporeality of hybrid Agents, whilst Knuth seeks a re-consolidation of a 'classical' formation of the computing self as Person and a normativization of the machinic under a paternal pedagogics. Dijkstra in his wariness of the machine, appears furthest from accepting the independent operation of the Agent, and in preferring the computer manual over the hardware appears to give primacy to the Subject. Yet there is also a sense of engaging with the limit of computation in Dijkstra that places the role of humour, and the comedic, in an entirely different position from that of Kaufman and Knuth and does not offer a simple resolution of Rotman's distribution.

142 Sohn-Rethel, Alfred, *Intellectual and Manual Labour: A Critique of Epistemology* (London: Macmillan, 1978).

143 For an overview, see Landauer, Rolf, 'Information is Inevitably Physical', in *Feynman and Computation: Exploring the Limits of Computers*, ed. Anthony J. G. Hey (Reading, MA: Perseus Books, 1999), 77–92.

144 Wiener, Norbert, *Cybernetics: Or Control and Communication in the Animal and the Machine* (New York: John Wiley & Sons, 1948).

145 Lefebvre, Henri, *Rhythmanalysis: Space, Time and Everyday Life* (London and New York: Continuum, 2004), 75.

146 Ibid., 40.

147 Ibid., 44.

148 See 'Mirror Neurons, Linguistic Negation, Reciprocal Recognition', in Virno, *Multitude* (emphasis in the original).

149 Black, D. W., 'Laughter', *Journal of the American Medical Association* 252(21) (1984): 2,995–8.

150 Copeland, 'From the *Entscheidungsproblem* to the Personal Computer and Beyond'.

151 The debates for and against recursion are discussed in Daylight, *The Dawn of Software Engineering.*

152 von Neumann, John, 'Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components', in *Automata Studies* (Princeton, NJ: Princeton University Press, 1956), 25.

153 Cantwell Smith, Brian, *On the Origin of Objects* (Cambridge, MA: MIT Press, 1996), 334–5.

154 Jagendorf, Zvi, *The Happy End of Comedy: Jonson, Molière, and Shakespeare* (Newark, NJ: University of Delaware Press, 1984), 17.