Fast Capitalism
Volume 15 • Issue 1 • 2018

ISSN 1930-014X
doi:10.32855/fcapital.201801.004

Writing Circuit Histories

Sascha Engel

1. Introduction

In addition to a substantial movie and TV industry, numerous recent books, both fiction and non-fiction, have discussed the end of man. To some extent, such cultural formations encode growing unease with the foundations of supposedly victorious Western society after the Cold War, reflecting the fragility of knowledge in the age of "fake news" on the one hand and increasing global uncertainty on the other. Nor is it an accident that such unease manifests in the form of engagements with the end of man. Beyond worrying about knowledge and its certainty, the end of man marks a contemporary formation of knowledge in its own right. Yet what is sorely missing is an appraisal of what empirical figurations may inhabit the space opened by the end of man, particularly in light of the unease it reflects. In contemporary Western society, where the "beasts and gods" between which man was situated for Aristotle have long since given way to human-machinic entanglements and human-animal hybrids, it is imperative to reassess the knowledge formations to which the end of man gives rise, and to ask what semi-human, semi-machinic assemblages inhabit them (Leslie 1996: 3-13).

It is the contention of this paper [1] that semi-human circuitries extend beyond formerly unified bodies and connect formerly distinct entities. In the section following this introduction, I argue that the end of man presents an opportunity to narrate histories of entangled circuits bridging human-animal-machinic divides. Buried beneath ontological distinctions separating man, machine, and beast, such forgotten histories must be told in ways that allow critiques of anthropocentric linear history. With this, mapping the terrain previously obfuscated by this linear historicity becomes possible.

In the third, fourth and fifth sections of this paper, some such critical mappings are presented. Their common theme questions one of the predominant linear narrative devices to which semi-human, semi-machinic assemblages are subject: the narrative of "progress." In the third section, I argue that a more richly socially embedded narrative is necessary to appraise exactly what it is that constitutes "progress." Particularly, I highlight the frequent appearance of retardation, delay, and hesitancy within the ostensibly unbroken march of progress.

In the subsequent fourth and fifth sections, I discuss the machinic side of such stories. Rather than following a trajectory of linear ascent – say, from less convenient to more convenient, from slower to faster, and so forth – machinic histories are likewise histories of dispersals, hesitations, and bifurcations. What emerges, therefore, are multi-faceted histories of semi-human, semi-machinic circuits, allowing critical engagements with linear historical narratives.

2. Circuit Histories

The transcendental condition of possibility for histories of circuits is the end of 'man' as an identifiable formation of knowledge. According to Michel Foucault's study on the Order of Things, "man" denotes a precisely dated formation of Western knowledge. It is preceded by knowledges based on similitude in the 16th century and succeeded in the 20th century by knowledges consisting of psychoanalysis (the other of conscious man), ethnology (the other of European man) and literature (the other of speaking man) (Foucault 1994: 42-44, 373-386). In the early

21st century, these three knowledges have in their turn given way to full eclipses of man at all sides: knowledges of stochastic distributions of catastrophe and accident; knowledges of environmental disaster and displacement; knowledges of undecidabilities and indeterminacies (see, for instance, Perrow 1999 or Lawrence and Wiebe 2017). Given the uncontrollable proliferation of the other-than-human, "[i]t is no longer possible to think in our day other than in the void left by man's disappearance" (Foucault 1994: 342).

Foucault thus reveals that "man" had at one point emerged as the condition of possibility of Western knowledge. By the same token, its role as gateway was bound to be finite. "Man" is not irreplaceable at the heart of knowledge; nor is anthropocentrism unavoidable. It is always already beset by its own dissolution: "at a very deep level, there exists a historicity of man which is itself its own history but also the radical dispersion that provides a foundation for all other histories" (ibid: 370).

The end of man uncovers this radical dispersion and provides the transcendental condition of possibility for different knowledges (Nietzsche 1989: 162-163). It opens pluralist and non-linear fields of histories, in contradistinction to the monomanic line of history prevailing as man held sway over Western knowledge formations (Chakrabarty 2007). Since man is dead – both in the temporal and in the conditional sense – life, labor and language inhabit fields of their own, with efficacies, distributions, and formations of their own (Latour 1993: 3). What is more, singular history gives way to plural histories as the dissolution of what was formerly "man" lays bare knowledges of multi-layered entanglements. Beyond Foucault's diagnosis, life now comes to be entangled in circuits of biotechnology and bioeconomy; labor comes to be stratified in circuits of cognitive and replaceable performance; and language scores and is scored in circuits of affect and social mediation (Galloway and Thacker 2007; Lovink 2011).

At this juncture, the historian is called upon to "draw a line around the short-lived facts" which, "[f]or contemporaries… hold the fascination of a fireworks display," and instead to focus on "[t]he constituent facts," which, "by accumulation and accretion… form the core of historical growth" (Giedion 1969: 389). Such are the constituent facts of our time that after "man's" death, writing history means writing history of the circuits in which life, labor and language are now mediated.

In these mediations, machines mingle with "humans" and "animals." Things have Internet, cameras are everywhere, and power is no longer based on macroscopic writing but on microscopic coding (Kittler 1993: 226). "In relation to objects like bionic components, one must think not in terms of essential properties, but in terms of design, boundary constraints, rates of flows, systems logics, costs of lowering constraints" (Haraway 2016: 30). One must write histories, not history, as objects no longer come to be constituted by one sovereign gaze in the cold light of unequivocality. Rather, objects become "constellations," and as such "readable as sign of their objectivity… Such constellations' being as writing is the transposition of that which is subjectively thought and brought together to objectivity through language" (Adorno 1975: 167-168). Rather than exhaustively described totalities, objects are now constituted in proliferating narratives amid fields of dispersing knowledges. Histories of these dispersals must be histories of semi-human, semi-machinic circuits, where it is "not clear who makes and who is made in the relation between human and machine" (Haraway 2016: 60).

A recent example for this is "Moore's Law," the assertion that the number of transistors in an integrated circuit doubles every other year, and that therefore processor capacity extends by roughly the same factor. When it was asserted in 2016 that "Moore's law has died at the age of 51 after an extended illness," the underlying narrative was one of human ingenuity and triumphant progress (Bright 2016). In all those years of hardware development, "life-changing things [had been] made possible by the reliable, exponential growth in the power of computer chips over the past five decades," such as "[m]obile apps, video games, spreadsheets, and accurate weather forecasts" (Simonite 2016).

Underneath this, however, a radical dispersal opens up as Moore's Law is reconsidered in light of the end of man. For one, technological advancement is here inextricably intertwined with economic calculus. It is thus hardly surprising to see that the beginnings of the end of Moore's law are primarily economic in character rather than technological: the end of Moore's law heralds the end of profitable processor capacity expansion – not expansion **per se** (Scientific American 2013). What is more, ontological boundaries evaporate in a circuit history reengagement with Moore's Law. For from constituting a history of ever-more extensive human advancement through technological progress, histories of processing capacity revolve around haphazard guesses, ideological commitment, Cold War politics of superseding socialism, private- and public-sector incentives, and so on (see, for example, Khan, Hounshell and Fuchs 2018).

In many ways, such ontological entanglement is not a new situation. Throughout "the traditions of 'Western' science and politics… the relation between organism and machine has been a border war" (Haraway 2016: 7). Much

of the critical edge of circuit histories arises from this conflict. Politics has always taken its place at this border where "[n]ature and culture are reworked" (ibid: 9), mixing and mingling to form "quasi-subjects" and "quasi-objects" equally as "unstable and hazardous" as "quite real" (Latour 1993: 89). As early as Marx, narrating history guided by a notion of "man's self-creation through labor" with an emphasis on the "historical process" by which "man" comes to constitute itself through natural and machinic circuits had come with a sharpened analytical eye for the political conflicts shaping the contested identities of man, nature, and machine (Rockmore 2002: 192). "For Marx, … humans and machines are continuous forces" (Wendling 2009: 118). On the one hand, "[i] n the historical and genealogical account Marx gives of machines, he shows that they are frozen labor of the past, and thus human and very political in content". Simultaneously, however, "humans, when portrayed in energetic terms, are machine-like" (ibid, 118-119). Circuit histories thus uncover the mixing and mingling of human and machinic ontologies, and the concomitant conflicts and losses buried underneath triumphant narratives of human progress and enlightenment. By traversing narratively constructed and upheld ontological boundaries, circuit histories show that the conflicts, losses, and horrors of history undermine the very boundaries they constantly re-erect to prevent being seen (Adorno 1975: 202-203).

3. Social Circuit Histories

The constituent facts of today, then, point to writing circuit histories rather than human history. Writing such circuit histories requires, first and foremost, assaying the canon of linear history seemingly removed from human intervention, and restoring the radical dispersal of constituent facts buried under the corpse of "man." Above all, an intervention is called for when it comes to the history of "technological progress." This is particularly necessary in the second decade of the twenty-first century, as contemporary "machines have made thoroughly ambiguous the difference between natural and artificial, mind and body, self-developing and externally designed, and many other distinctions that used to apply to organisms and machines" (Haraway 2016: 11).

Here in particular, standard historicity based on the "progress" paradigm ignores that histories are nonlinear and entangled, and that "invention," "improvement" and "innovation" never come without social context. Identifying and discussing technological artifacts as steps in an evolutionary paradigm of progress and amelioration almost inevitably naturalizes the social mediation of technological developments. At a time when "[o]ur machines are disturbingly lively, and we ourselves frighteningly inert" (Haraway 2016: 11), specters of progress present humanity either as enlightened helmsman or as victim of anonymous processes. Both of these narratives ignore the social nature of what they present as an irreversible development of means and modes of production. On the one hand, presenting fields of exemplary progress such as Artificial Intelligence as a result merely of human perfectibility and ingenuity conveniently forgets that machinic histories have rules of their own and form fields of dispersion of their own. The system of machinic objects contains complexities and spillovers, bleedthroughs and externalities. "Artificial Intelligence" in particular spans a wide field where the androids of corporate capital and state warfare dream of electric surveillance and displacement of labor with the same intensity with which they wage war against one another. Ignoring the constraints to which any algorithm is subject in a Gödelian universe, such fever dreams are predicted upon a "mystique of information that makes basic intellectual discriminations between data, knowledge, judgment, imagination, insight, and wisdom impossible" (Roszak 1994: xix). Papering over any misalignment, setback, or conflict, "Artificial Intelligence" embodies "progress" like no other paradigm in the early 21st century.

On the other hand, humans are not mere electric sheep in the face of machinic menace and mayhem. This, too, is too simplistic a narrative. It particularly – and quite conveniently – forgets that technological development is hardly foreign to "a context that includes relative prices, regulatory and other institutional factors and, obviously, the perceived market potential of the innovations concerned" (Perez 2010: 186). The effects of the improvement of processing speed encapsulated in Moore's Law, for instance, are clearly socially stratified. Consider, for example, the replacement of workers in the fast food industry with automated check-out points – while, simultaneously, outsourced customer service presents consistent employment growth predicated upon the very same human interaction which fast food chains evidently no longer require (Bureau of Labor Statistics 2018). Such differentiated effects, too, must be described in detail: gains for some, losses for others.

Consequently, writing circuit histories must here primarily be critique. Any time differentiated accounts of social gain and loss are papered over by progress narratives, social phenomena are reified into natural phenomena. Everyday

life, labor and language thus come to be subjected to a technological paradigm elevated beyond question (Lefebvre 2002/1961: 74-78). Moreover, this self-evident paradigm is inherently totalizing (Siegel 2008: 27). Empirical fields of pluralist histories thus come to be lumped into a natural history of ever better and ever more comprehensive technocratic solutions to social problems (Ellul 1964: 116-133).

Thus, writing circuit histories means first and foremost engaging in a sustained critique uncovering the social cost of so-called "progress" (Haraway 2016: 37). It should hardly require pointing out – but all too frequently does – that not all new technologies are also improvements. The notion of "progress," however, removes the means to properly evaluate new tools and gadgets (Siegel 2008: 18). Moreover, it precludes the recognition of legitimate critiques of the social effects of the introduction of new technologies. As the fate of the "Luddites" in particular shows, any social movement opposing even parts of technological implementation is susceptible to being vilified – or worse, ignored – by narratives of "progress:"

> No one alive today remembers firsthand the trauma that we call the first Industrial Revolution... The inherited accounts of this period were formulated by and large in response to the dramatic actions of those who fought for their survival against this progress. They constituted a post hoc effort to deny the legitimacy and rationality of such opposition... The Luddites... did not believe in technological progress, nor could they have; the alien idea was invented after them, to try to prevent their recurrence (Noble 1993: 4).

Even beyond social cost, and only looking at the – as it were – positive side of a "progress"' balance sheet, one will not uncover linear ascent. Taking a closer look at the actual histories of "progress" rather invites comparison to the succession of paradigms in Thomas Kuhn's study of science (Wojick 1979: 238). For Kuhn (2012/1962), scientific progress occurs by briefly punctuating long periods of unquestioned scientific normality with rapid overhauls of the paradigms upon which this scientific normality had been predicated. In the implementation of scientific progress into everyday engineering, "received evaluation policy … plays a role analogous to that played by an accepted paradigm in an area of scientific explanation" (Wojick 1979: 244). When a new technology arises which "enables us to see that our standard procedures do not evaluate all factors correctly," the new paradigm "may lie outside the group or discipline in charge," or worse, "the evidence for anomaly or misevaluation may be tentative, controversial, or merely qualitative" (ibid, 245). For example, one of the results of the development of Artificial Intelligence seems not so much to have been the success or failure of specific machinic entities, but that it put established measurements of "intelligence" in question. Particularly, failures of the so-called Turing test, where human operators are supposed to find out whether they are conversing with a machine or a human, have raised doubts regarding methodologies of measuring intelligence (Batson 2014).

Yet, as "progress" marches on, advocates will split from conservatives, and initially unclear positions on both sides will result in conflict. Once this conflict goes public, progressive "popularizers" split from conservative "technologists" (Wojick 1979: 246). Here, too, Artificial Intelligence provides ample examples for both sides of this debate. For instance, the recent feud between Tesla CEO Elon Musk demanding further government regulations for the use of Artificial Intelligence, and Facebook CEO Mark Zuckerberg emphasizing market-based innovation, exemplifies that conflict between the popularizers and technologists arise in any field of technologically accentuated development (Solon 2017).

The "technologists" frequently have the upper hand initially since "the lay public may not appreciate the differences between the crude new evaluation policy and the well-articulated established policy" (Wojick 1979: 246). Consequently, "progress" narratives tend to prevail with ill-informed audiences. The adjustment period, in turn, will be publicly interpreted as confusion or even "steps back;" particularly if it brings grave social consequences (Noble 1993: 5-6). In the end, however, "[t]he confusion cannot last," and the new policy prevails (Wojick 1979: 259). Its social consequences are then pushed aside by various non-violent and violent means until the mere idea of opposing the new paradigm comes to be seen as irrational or deluded (Noble 1993: 16-17). Artificial Intelligence has not reached this point yet, but already its proponents accuse its opponents of obscurantism (Walker 2017). Likewise, critical assessments of Moore's Law, particularly those showing its economic rather than technological nature, have historically been met with hostility (Ceruzzi 2005: 590).

Even in a highly idealized form, then, histories of socially mediated technological dispersal are histories of confusion, misalignment, personal conflict, and ill-informed intervention rather than linear ascending pathways. Even in a petri dish, progress is political. Beyond such idealized circumstances, moreover, histories of technological dispersal will have to take economic and political agendas into account, as well as ideological and plain old pork barrel politics (Kellner 1992: 187). Messy though they are, these are nevertheless the constituent facts of techno-social

"progress."

4. Machinic Circuit Histories

Just as circuit histories uncover that each individual affected by "progress" is a history unto itself, thus humanizing the machinic, so they also bring micronarratives into play by which the machinic intersects with the social. A history of the entanglement of hitherto "human" and hitherto "machinic" circuits cannot therefore be content with just the critique of linear history. It must also pay close attention to the dynamics within the circuits it describes, and particularly to those within the machinic realm itself, overdetermining the conditions of entanglement of social and technological realms (Kellner 1992: 178). Reconsidering their object, circuit histories reject sweeping macronarratives in favor of microlevel precision. Just as "progress" history dissolved into small-scale narration of knowledge politics, so formations like "Artificial Intelligence" dissolve into constituent gestures, each of which has histories of its own, to be narrated on the micro-scale of its dispersal.

Such micronarratives augment, situate, and embed macronarratives as constituent facts disperse linear monomania. Thus, once again, narrating an exemplary field of such dispersed circuit histories constitutes a critique of an all-too-linear knowledge formation where hitherto the narrative of "progress" held sway. At the same time, micronarratives of machinic histories go against the grain of limited econocentric readings of history, focusing instead on machinic momentum. In this way, they round out the above socially embedded histories of machine development by adding a machinically embedded field of histories of social development.

The particular example discussed in this section demonstrates, moreover, the radical dispersal at work in the array of machinic figurations which can be uncovered beneath "progress." Exemplifying that dispersal lies not just underneath the linear timeline of progress, but also the unified object of "Artificial Intelligence," this section focuses on the histories of "loading." Big-picture items such as the question of machinic consciousness easily obfuscate that such consciousness, even if it were an attainable reality, would still consist of myriad tiny gestures. One of these is inevitably the question of loading – consciousness, after all, requires extensive initialization in human circuits as well. Loading denotes the act of initializing all values of a program to be executed: intermediary storage positions, GOTO loops, initial values (including terminal-based input), and of course the program and its associated subroutines themselves. Thus, loading is the process by which a Turing machine's initial 0-state is set up which is necessary for algorithm execution (Denning, Dennis and Qualitz 1978: 483).

Importantly, this refers both to the initialization of hardware and that of a coded routine. In FORTRAN, for example, "[t]he code that is to be executed must first be loaded into memory using the LOAD routine. This code to be executed is assembled and linked into an assemblage the LOAD routine will handle" (Kettleborough 1985: 184). Further down coded hierarchy, opcode and the parser themselves have to be loaded; a process repeated every time computing hardware boots up und streams of electric pulses manifest to UEFI, kernel, operating system, and eventually applications.

As is immediately evident, the multifaceted taxonomy of loading suggests multiple histories based on a variety of definitions. Still, an approximation to the concept is possible. In both the coded and the hardware version, loading is distinct from compiling, which generates the structure by which loading occurs (Backus et al 1957: 26-27). Likewise, it is distinct from the manual entry of initial values, to which it rather assigns intermediary storage space as structured by the compiler (Booth and Booth 1965: 222-223). Loading is somewhat closer to gestures such as memory dumps, where values are retrieved from storage locations and loaded into the present routine (IBM 1974: 116-119). Likewise, loading bears some kinship with diagnostics, where value initialization is implemented for testing purposes (Kettleborough 1985: 62-64).

Loading is a particularly good example for a critique engendered by writing circuit histories because of the monolithic and teleological character of its prevailing narrative. In 21st-century program initialization, loading is essentially invisible. This follows partly from processing speed, closely aligned with aspects of commercialized convenience. In the graphic interface representation upon which the vast majority of contemporary operating systems are predicated, loading is at best a nuisance papered over by introductory graphics. At worst, it constitutes a fatal problem, as was the case recently with LG Nexus devices and their Oreo update.[2]

In the linear teleology of "progress," the disappearance of loading is thus an ideal end state, of which all previous loading routines are imperfect approximations. From its inception, graphics interface software was written

with the aim that it "should be easy to use but at the same time provide as many useful features as possible," including compatibility in output with "lower quality devices" - a feat direly needed as state-of-the-art interfaces remained plagued by slow speeds and thus long loading times (Sutcliffe 1980: 52). What relief, then, that Windows 10 which already "came with no shortage of performance improvements" presents as its "neatest" feature "its fast booting times" (Ravenscraft 2015)! Likewise, reducing speed constitutes progress in hardware processing capacities to such an extent that the ideology of Moore's law gave way to "Meltdown" and "Spectre" in January 2018, two processor vulnerabilities exploiting a time-saving technique used in bootloading contemporary operating systems.

Yet, the histories of semi-machinic circuits inhabiting the terrain between compiling, retrieving, dumping, and testing hardly give credence to a narrative of ever better hidden loading procedures resulting in ever-improving speed and ever more convenience. Rather, circuit histories trace a dispersion of gestures resulting in histories of displaced human-machinic interaction, particularly centered around power differentials encoded in hardware and software access levels.

In 1957's TYDAC, for instance, loading was a largely hands-on affair. It was initialized, first, by a direct and manual choice between different inputs in the following command:

60 SELECT x

where "x" is a selection from the set of possible input channels: 1 addresses the Card Reader, 11 does so for Tape unit 1, 12 for Tape unit 2, 13 for Tape unit 3, and 14 for Tape unit 4. After initializing this choice, a second command sets in motion the actual loading process:

61 READ 1000,1

where 1000 is the address into which the tape's content is to be read, and 1 is the index register to be used for the operation (McCracken 1957: 220).

The constituent facts contained in this initial loading gesture are those generally found throughout later versions of loading as well: the choice of input from which loading occurs; setting the storage addresses in which data is received; temporary storage; the material act of data transfer itself. Three aspects of this are nevertheless remarkable. The first of these is that setting the temporary storage address is not done by a compiler routine but is set manually at initializing the loading command. Here, encoded access is total inasmuch as the code addresses all input channels equally. Secondly, however, to a significant extent this is due to limitations in loading structure. In TYDAC, as in late-1950s machines more generally, loading primarily operates by feeding card stacks (or tape) into a reader. Automating this, in turn, still presupposes direct interaction, as the user "must somehow get the first card in, which must have on it a program which will load the remaining cards" (McCracken 1957: 142-143).

Thirdly, the choice of input device is not directly part of the loading gesture. This is particularly intriguing here since it directly integrates not just input, but also output into loading gestures. The command selecting inputs

60 SELECT x

addresses two other options as well; both of which are outputs. Entering "2" selects the Card punch, while "3" addresses the Typewriter (McCracken 1957: 220).

Contrary to expectations of "progress," one thus finds more integration and greater degrees of user interaction here than one does in later incarnations of loading routines. Particularly once loading – both booting for operating systems and loading for individual programs – came to be hidden behind graphic interfaces, such choice and interaction all but evaporated.[3]

Thus, for example, 1981's Sinclair ZX81 exclusively features a tape loading routine. Furthermore, apart from winding the tape to the program's starting point and connecting the sockets, manual user interaction with the hardware disappears behind a graphic veil. So does most of the coding. Typing LOAD, without any qualifiers as regards source or final location, starts the tape's input, whose only immediate hardware stipulation to be heeded by the user is that the tape be regulated tonally: "maximum treble, minimum bass." Once loading is initiated, "you will see various fairly even patterns on the screen, and then suddenly a rapidly moving pattern of horizontal bars… This is your program. After loading, the screen will clear with a 0/0 report code" (Norman 1980: 58).

User interaction with the loading process is reduced to setting up timings and time stamps on the tape in question and winding the tape up to the exact starting point (Norman 1980: 57-58). The tape's tonal regulation further qualifies the reference frames of intelligibility for the magnetic pulses to be derived from the tape (ibid). Everything beyond this threshold is devoid of direct interaction and encoded to prevent access: the "various fairly even patterns" are the tape's sounds while they still remain just unrecognized sounds, while the subsequent "rapidly moving bars" represent those same sounds, transformed once a threshold of intelligibility is crossed – once the program is identified as a program.

The user thus chooses a fixed program, to be relocated from one fixed location on tape to another in main device storage. Hiding the realities of loading behind this veil of seeming transparency is, not least, commercially relevant, as only the reification of input and output allows its packaging in "programs" and – eventually – "applications." Were choices left to the user, such regulation of choice and creativity to that between various types of commercially available products would be threatened. Yet what is hidden here exceeds such immediately commercial considerations.

Rather, commercial reification at play here is part of a larger displacement. Up to a certain point, the initial sound read from ZX81's tape is indistinguishable from noise. It is only when that point is reached and its threshold crossed that the "moving bars" retroactively establish the intelligibility of the previous patterns and the program's phantasmagoric objectivity arises. Code, as an emergent quality, encodes its own intelligibility threshold. In turn, this threshold delineates access. On the one hand, user interaction must be denied initially as the very intelligibility of user interaction must first be loaded. On the other hand, cutting off user access in this way prevents any program other than the present one from being initialized. Commercial reproduction and the delineation of intelligibility go hand in hand.

As regards the coded emergence of code itself, consider this description of loading hardware from the 1960s:

> **...a group of order digits is just appearing at the digit output of M[emory] and... the binary element B is set so that the gate g1 is open and the train of 32 clock pulses passes through to C[ontrol] R[egister]. These pulses cause the contents of C.R. to shift progressively to the right and, at each stage, one of the incident digits from M is absorbed. When the whole 32 have appeared these will be stored in C.R. and the memory emits an operation complete pulse which is incident upon the right-hand input of the binary element and causes a state change so that g1 closes... (Booth and Booth 1965: 35).**

Here, electromechanical input operates directly on the hardware level. The opening of a gate is followed by clock pulses regulating the main transmission of actual pulses ("incident digits") from a binary input source to a register storage where storage is implemented via right shift. In turn, this is followed by a pulse indicating the end of transmission and thus closing the gate.

Here, it is particularly remarkable that the incident digits – that is, the actual values to be loaded – remain secondary to the transmission of clock pulses. To the computing device, the important part of any data transmission – including loading routines – is the synchronization of clock pulses between peripheral and mainframe sectors (Phister 1960: 175-178). This is congruent with the characteristics of a Turing machine. After all, the origin of Turing machines – quite removed from their contemporary usage – is the establishment of a mathematical boundary of computability in a thought experiment (Denning, Dennis and Qualitz 1978: 489). From the inception of computing, "content" had been secondary, to the point of irrelevance. One of the histories of loading, then, is the history of an inertia, as this particular formation remained at the center of loading gestures ever since. "Progress" is notably absent in this regard.

A particularly good example of this is the loading gesture in 1967's PDP-8/I. Once tape loading was initialized, establishing communication between DECtape and the PDP-8/I main device consisted, firstly, of setting error flags synchronizing the tape's reading sequence with the main device's status register (Digital Equipment Corporation 1967: 341).[4] A second element at the heart of PDP-8/I's tape loading process was the synchronization of tape speed with that of the main device. This is why "timing and mark channels are recorded prior to all normal data reading and writing on the information channels," and indeed is why, as the user is brusquely informed, "[s]oftware supplied with DECtape allows writing for fixed block lengths only" (ibid: 184, 189).

Here, too, technical and commercial quasi-necessity go hand in hand. Removing content from the equation of computing – both literally and figuratively – serves to consolidate the process of loading itself. Since loading is exclusively about synchronization, the device from which data is loaded and the mainframe upon which data is to be stored must be fundamentally compatible. The easiest way to establish such compatibility, of course, is to base the process of loading exclusively upon proprietary hardware. By the same token, the operational establishment of synchronization functions most easily if it is automated – and that is, if user access is removed in the process

of loading itself. Thus, DECtape only works with DEC devices and those devices for which DEC has established compatibility, and the user of a ZX81 will try in vain to read a data tape from the TYDAC era into her device. Nor, on the other hand, does this commercial encoding follow directly from machinic synchronization: it is entirely conceivable, absent commercial imperatives, that device synchronization can bridge the divides of proprietary hardware. Indeed, it frequently does – except when it is prohibited from doing so.

5. Two Further Threads

A third history of loading is that of a transition from physical to logical tape input and back (Digital Equipment Corporation 1967: 185-188). This physical/logical differentiation is encoded in an exemplary fashion in FORTRAN. Reminiscent of TYDAC, FORTRAN's READ is a general input command following up on another command structuring the way data is received. In FORTRAN's case, this is accomplished by a FORMAT code (Backus et al 1957: 26). As in TYDAC, this FORMAT instruction contains a choice of input type. Unlike in TYDAC, it then specifies the exact floating-point numbering format for data storage (ibid: 27). It is noteworthy that FORTRAN also contains output structuring code, such as carriage control characters: "blank" for a single space, "0" for a double space, and so on (ibid: 29). With these, input and output are on their way to becoming internal parts of the program itself. This step is ambiguous: on the one hand, it sets up the path towards the mystifying loading screen implemented in Sinclair's ZX81 by internalizing hardware access into its encoding. In doing so, however, FORTRAN also allowed more direct access to setting up the way data is received. The interface is encoded, but not entirely hidden.

With this, a bifurcation occurs in the histories of loading. PDP-8/I's and FORTRAN's differentiation of pure input from structured input, which is replicated in ZX81's difference between pure sound and intelligible sound recognized as input, delineates a distinction of loading within a graphic interface – such as a present-day operating system – from loading that operating system itself. Loading is now increasingly split between bootloading and program loading. Where TYDAC's loading procedure offered direct addressing of hardware, and even ZX81's graphic encoding left some hardware elements intact, this split now removes those last traces. Some twenty years after PDP-8/I, consequently, C64's LOAD is a high-level command setting a subroutine in motion whereby "consecutive data bytes" are moved "from input device to Commodore 64 memory" (Philipps, Nath and Silveria 1984: 73). At that stage, loading exclusively refers to program loading and is entirely distinct from booting.

With this displacement of hardware access, two secondary access levels are distinguished: loading in the booting sense now comes to be associated with external storage media, while loading in the program loading sense is situated within the main device. This makes sense because bootloading installs the first layer of hardware obfuscation, the operating system. On its basis, in turn, individual loading routines can encode and effect the reification of "applications." An exemplary manifestation can be found in IBM's System/360 family, about ten years after the PDP-8/I, where loading is transformed into an internal function within assembly routines. It no longer refers to input loading from tape or drum, but rather loads index registers (Opler 1966: 39).

Nevertheless, hardware obfuscation or displacement remains negotiated and uneasy. Once completely internalized, the programmed loading gesture uneasily re-incorporates direct input and output accessing in the 80386 processor family of the late 1980s. Here, loading oscillates between operating system and application in the "Input from Port" command. A return to TYDAC's choice of input seems to be preserved, as it is possible to "access any port from 0 to 65535 by placing the port number in the DX register and using an IN instruction with DX as the second parameter" (Intel Corporation 1987: 17/65). This access remains at the processor level, however, and does not concern the operating system itself.

What is more, a parallel to ZX81's loading screen returns, removing user access even at the processor level, albeit in a different way. While the 80386 instructions seem to restore input/output port choice to the user, direct addressing of peripheral devices as in TYDAC is nevertheless precluded by access constraints. In 80386 opcode, an input/output access command must come from a specific privilege level, as "[a]ny attempt by a less privileged procedure to use a sensitive instruction results in a general protection exception" (Intel Corporation 1987: 8/5).

A fourth history of loading emerges here. Thus far, direct loading or bootloading and indirect loading or application loading have been distinguished as two levels. With the installation of opcode privilege, a third layer of encoding emerges. Moving from direct loading to this new level, the scope of loading expands markedly: it includes in the 80386 family the loading of effective addresses (Intel Corporation 1987: 17/91) as well as status words

(ibid: 17/99) and strings (ibid: 17/102-103). With the internalization of loading effective addresses in particular, the mechanical act of winding up the memory tape to its starting point, a core element of early loading, now becomes internal to the program. Synchronization between mainframe and periphery takes on an expanded form, now established not in the bootloading process, but in the program's own opcode. In turn, this is implemented by the assembler, converting relative addresses to absolute addresses and thus ensuring the synchronization of application and operating system as well as mainframe and periphery. The now doubly encoded loading routine is thus further removed from the user still, and an equivalent of ZX81's loading screen or the 80386's privilege level architecture removes access entirely.

Even before the 80386 family emerges, IBM's System/360 family implements this further displacement in its assembler routines (IBM 1974: 20). At the beginning of each program to be executed, a USING command sets up the base register, tying the program's internal relative addresses to an absolute position in the device's memory space (ibid: 51). Subsequently, a BALR command stands "at the beginning of a program… getting the address of the next sequential operation from the current program status word, no matter where the program may have been located" (ibid: 24). Beyond allowing relative addressing within the program, this also reifies the program as such by establishing dynamic anchoring of programs within absolute memory space. Loading is no longer a universal command to synchronize any device with its input/output ports and thus, secondarily, to load any content. It has rather come to be part of a specific routine, loading specific content for specific purposes. At the same time, the USING command removes the choice of where to store a program in absolute memory space from the user and implements it in the assembler. Removed from direct access by the layers of "application," "assembler," and "operating system," the user stares at the equivalent of ZX81's loading screen until something happens. The "application" can freely be implemented as a commercial entity.

It is hardly surprising that this development culminates with a second-order encoding of loading itself. In System/360, just as in ZX81, loading becomes a predefined routine, available to the assembly coder – never mind the end user – merely as a pre-structured macro (IBM 1974: 123). Entering LOAD in the assembler's code "obtain[s] a full word (four bytes) from storage at the effective address specified, and place[s] the word in the general register indicated" (ibid: 30). Likewise, a multi-LOAD variation of the same macro "begins loading fullwords from the specified storage location. The first word goes into the first-named register. Successive fullwords go into higher-numbered registers until the second-named register has been loaded" (ibid: 40).

The displacement of the user from direct access to hardware and ultimately from loading itself is thus complete. In 1988's APL2 language, to use just one example, loading consists in defining a virtual vector, such as an array, in which items are to be stored; the definition of a relative storage address where they are to be stored; and a definition of the "stride", ie, the virtual distance between subsequent elements (Brown, Pakin and Polivka 1988: 350). The loading routine is here still initiated directly by the user, and the storage address – albeit not the absolute one – is still set manually. Yet, storage formats are predefined and the assembly of this command is as much out of the hands of the user as the definition of an array is out of those of a 21st-century "developer" in contemporary front-end "web design."

6. Conclusion

The space of knowledges left in the wake of man's end opens avenues of critical examination. Circuit histories exploring these avenues, first and foremost, uncover dispersed elements hitherto buried underneath linear histories of "progress." By the same token, they show that seemingly monolithic entities such as Moore's Law or Artificial Intelligence consist of myriad smaller formations with their own histories, like those of loading and initializing on the machinic end; quarrels between CEOs and differentiation between strata of workers on the human end. Underneath the social history of human innovation and ingenuity, circuit histories uncover subterranean pork barrel politics and knowledge regimes as well as hesitation and retardation in assaying and applying innovation. The partly social, partly machinic formations of "progress" thus remain embedded in a messy and complex, multidimensional reality without fully realizable teleology. Taking "man" out of historical narratives allows taking stock of displacement rather than teleology. Quasi-laws like Moore's render invisible the economic and cultural factors leading to their seeming certitude. Artificial Intelligence occludes a vast field of conflicting interests and uncertain applications, as well as differentiated gain and loss. Both paper over human experience.

By the same token, circuit histories of machinic formations show that social effects arise as much from machinic formations as vice versa. Power differentials are encoded and decoded in series of hardware and software displacements uncovered only when the narrative history of progress gives way to narratives emphasizing histories of dispersal. What is encoded in the histories of code is not a teleological move towards ever more convenience and transparency in service of users. Rather, ever increasing layers of access removal from users encode reifications of operating systems, assemblers, and applications. In turn, this feeds back to commercial forces. Particularly the development of internalized memory address initialization culminating in System/360's USING command allows the reification of programs as products. Such "applications" can then be anchored anywhere within absolute memory ranges. Variability of this kind allows copying and selling the commercial application en masse and independent of the device in question. In turn, the latter can be reified to "operating systems." On either level, this only works if loading is further reified. In this circular movement feeding back into itself, removing user access to bootloading routines allows a reification of operating systems, along with applications, as self-contained economic entities.

At the same time, it is important to remember that encoded social and economic imperatives remain subject to the machinic dynamics of both code and hardware. After all, the shift from System/360's loading routine to that of the 80386 family does re-enable some direct access of ports. On the other hand, this restored degree of choice remains within the assembly level, removed from those users whose access or knowledge does not extend that far. Even so, it is further removed still by a second-order imposition of privilege levels within the 80386 architecture. As part of such movements from the machinic to the social and back, too, the mass marketing of programs and "applications" emerges from the architecture of loading routines as one possibility of reifying the transition from relative to absolute memory indexing.

What emerges, then, is a complicated picture in which neither economic nor technological rationalities hold sway entirely, and in which messy human politics are just as effective in encoding, recoding and decoding the things to come as are machinic assemblages. There is no linear "progress" in the politics of social interactions, because the perils and platitudes of scientific popularization and engineering applications, along with market miscalculations, personality clashes, and coded access politics prevent or displace invention and innovation just as much as they further it. Nor is there "progress" in the trajectories of machinic lineage, as machinic assemblies are constituted more by lateral movements, dispersals, and displacements, than by innovative amelioration.

What emerges, in other words, are fields of dispersals instead of linear narratives, and crisscrossing ontologies instead of "man's" dominion over the sequence of time. Writing circuit histories gives voice to such dispersals, en route to critical reappraisals of man-machines, their natures, and their constellations. In the present state of anxious uncertainty, unsettling all-too-easy narratives can thus contribute to avoiding rash judgments. Once real existing non-progress is exposed, the notion might lose some of its status as panacea for the ills of the world. At the same time, the real effects papered over by its narrative triumphalism, from circuit access exclusion to joblessness, can be brought to light. Particularly, circuit histories expose the neutrality of code as a myth, arming critical analyses of power with further tools to decode the minutiae of social silicone and machine marketing.

Endnotes

1. It could not have been written without the generous inspiration and encouragement I received from Eoin Murphy. To him, Tim Luke, and the anonymous reviewer of this paper I owe a debt of gratitude.

2. This update was halted after it emerged that it damaged some phones' booting procedures to such an extent that they effectively became unusable. At the time of writing this, the exact cause(s) appear(s) to be a bit of a mystery still, but it seems that random forced reboots, battery problems, as well as memory issues were to blame.

3. It is intriguing that TYDAC, with its strong focus on the matter of hardware, was in fact a "mythical computer," the "TYpical Digital Automatic Computer... intended primarily as an aid to learning" (McCracken 1957: v). The history of actually existing computing begins with many such ficitious entities: neither TYDAC nor the Analytical Engine were ever materially implemented, while Alan Turing's initial concept of a Turing machine was a thought experiment designed to address one of Hilbert's problems. Here, too, social factors are important, as a significant part of the secrecy surrounding Turing's work in particular is related to the Cold War. While this matter – or rather, the absence thereof – remains beyond the scope of the

present paper, the ghostly character of early computing hardware in particular certainly merits more attention.

4. To be precise, this consisted of two actions: the DTCA command, clearing the status register of the main device, and the DTXA command, setting the status register according to the reading from the DECtape control flag. Here, one could branch off an exploration of the microhistories of synchronization, from tape and drum memory time stamps to present-day latency reduction.

References

Adorno, Theodor. 1975. Negative Dialektik. Frankfurt: Suhrkamp. My translation.

Backus, J., R. Beeber, S. Best, R. Goldberg, L. Haibt, H. Herrick, R. Nelson, D. Sayre, P.

Sheridan, H. Stern, I. Ziller, R. Hughes and R. Nutt. 1957. The FORTRAN Automatic Coding System for the IBM 704 . Los Angeles: Proceedings of the Western Joint Computer Conference, pp. 188-198.

Batson, Joshua. 2014. "Forget the Turing Test: Here's how we could actually measure AI," Wired. Accessed 2.22.2018 at https://www.wired.com/2014/06/beyond-the-turing-test/

Booth, Kathleen and Andrew Booth. 1965. Automatic Digital Calculators. London: Butterworths.

Bright, Peter. 2016. "Moore's law really is dead this time." Ars Technica. Accessed 2.18.2018 at https://arstechnica.com/information-technology/2016/02/moores-law-really-is-dead-this-time/ .

Brown, James, Sandra Pakin and Raymond Polivka. 1988. APL2 At A Glance. Englewood Cliffs: Prentice-Hall Publishing.

Bureau of Labor Statistics. 2018. "Customer Service Representatives," Occupational Outlook Handbook. Accessed 2.18.2018 at https://www.bls.gov/ooh/office-and-administrative-support/customer-service-representatives.htm#tab .

Ceruzzi, Paul. 2005. "Moore's Law and Technological Determinism," Technology and Culture , 46 (2005), pp. 584-593.

Chakrabarty, Dipesh. 2007. Provincializing Europe: Postcolonial Thought and Historical Difference . Princeton: Princeton University Press.

Denning, Peter, Jack Dennis and Joseph Qualitz. 1978. Machines, Languages, and Computation . Englewood Cliffs: Prentice-Hall Publishing.

Digital Equipment Corporation. 1967. Small Computer Handbook. Maynard, MA: Digital Equipment Corporation.

Ellul, Jacques. 1964. The Technological Society. New York: Vintage Books.

Foucault, Michel. 1994. Order of Things. New York: Vintage Books.

Galloway, Alexander and Eugene Thacker. 2007. The Exploit. Minneapolis: University of Minnesota Press.

Giedion, Siegfried. 1969. Mechanization Takes Command. New York: W.W.Norton & Co.

Haraway, Donna. 2016. Manifestly Haraway. Minneapolis: University of Minnesota Press.

IBM. 1974. A Programmer's Introduction to IBM System/360 Assembler Language. White Plains: International Business Machines.

Intel Corporation. 1987. 80386 Programmer's Reference Manual. Santa Clara: Intel Corporation.

Kellner, Douglas. 1992. Critical Theory, Marxism, and Modernity. Baltimore: The Johns Hopkins University Press.

Kettleborough, Ian. 1985. Utah Fortran. Reno: Ellis Computing.

Khan, Hassan, David Hounshell and Erica Fuchs. 2018. "Science and research policy at the end of Moore's law," Nature Electronics, 1 (2018), pp. 14-21.

Kittler, Friedrich. 1993. Draculas Vermächtnis. Technische Schriften. Leipzig: Reclam, 1993.

Kuhn, Thomas. 1962/2012. The Structure of Scientific Revolutions. Chicago: University of Chicago Press.

Latour, Bruno. 1993. We Have Never Been Modern. Cambridge, MA: Harvard University Press.

Lawrence, Jennifer and Sarah Marie Wiebe (eds): Biopolitical Disaster. New York: Routledge.

Lefebvre, Henri. 1961/2002. Critique of Everyday Life. Vol II: Foundations for a Sociology of the Everyday . London: Verso.

Leslie, John. 1996. The End of the World. London and New York: Routledge.

Lovink, Geert. 2011. Networks Without a Cause. Cambridge: Polity Press.

McCracken, Daniel. 1957. Digital Computer Programming. New York: John Wiley & Sons.

Nietzsche, Friedrich. 1989. On the Genealogy of Morals. New York: Vintage Books.

Noble, David. 1993. Progress Without People: In Defense of Luddism. Chicago: Charles H. Kerr Publishing.

Norman, Robin. 1982. ZX81 Basic Book. Indianapolis: Howard Sams & Co.

Opler, Ascher (ed). 1966. Programming the IBM System/360. New York: John Wiley & Sons.

Perez, Carlota. 2010. "Technological revolutions and techno-economic paradigms," Cambridge Journal of Economics , 2010 (34). Pp. 185-202.

Perrow, Charles. 1999. Normal Accidents: Living with High-Risk Technologies. Princeton: Princeton University Press.

Phillips, Gary, Sanjiva Nath and Terry Silveria. 1984. The Commodore 64 User's Encyclopedia . Los Angeles: The Book Company.

Phister, Montgomery. 1960. Logical Design of Digital Computers. New York: John Wiley & Sons.

Ravenscraft, Eric. 2015. "Enable This Setting to Make Windows 10 Boot Up Faster," Lifehacker. Accessed 1.19.2018 at https://lifehacker.com/enable-this-setting-to-make-windows-10-boot-up-faster-1743697169.

Rockmore, Tom. 2002. Marx After Marxism. Malden, MA: Blackwell.

Roszak, Theodore. 1994. The Cult of Information. Thousand Oaks: University of California Press.

Scientific American. 2013. "End of Moore's Law: It's not just about physics," Scientific American. Accessed 2.18.2018 at https://www.scientificamerican.com/article/end-of-moores-law-its-not-just-about-physics/.

Siegel, Lee. 2008. Against the Machine. Being Human in the Age of the Electronic Mob. New York: Spiegel & Grau.

Simonite, Tom. 2016. "Moore's Law Is Dead. Now What?" MIT Technology Review. Accessed 2.18.2018 at https://www.technologyreview.com/s/601441/moores-law-is-dead-now-what/ .

Solon, Olivia. 2017. "Killer robots? Musk and Zuckerberg escalate row over dangers of AI," The Guardian. Accessed 2.18.2018 at https://www.theguardian.com/technology/2017/jul/25/elon-musk-mark-zuckerberg-artificial-intelligence-facebook-tesla .

Sutcliffe, D. C. 1980. "Contouring over rectangular and skewed rectangular grids – an introduction," in: K. W. Brodlie (ed) Mathematical Methods in Computer Graphics and Design. London etc.: Academic Press. Pp. 39-62.

Walker, James. 2017. "Mark Zuckerberg and Elon Musk: Does either really understand AI?" Digital Journal. Accessed 2.18.2018 at http://www.digitaljournal.com/tech-and-science/technology/mark-zuckerberg-and-elon-musk-does-either-really-understand-ai/article/499234.

Wendling, Amy. 2009. Karl Marx on Technology and Alienation. New York: Palgrave Macmillan.

Wojick, David. 1979. "The Structure of Technological Revolutions," in George Buglirello and Dean Doner (eds), The History and Philosophy of Technology. Chicago: University of Illinois Press.