To execute, rewrite, and debug

On the construction and deconstruction of computation

Gauthier, D.

Summary

To Execute, Rewrite, and Debug: On the Construction and Deconstruction of Computation

Current discourses on digital media display a fascination with the linguistic, numeric, algorithmic, social, and material aspects of technology, from which several new domains of study have emerged in the past several years, including Digital Methods, Software Studies, Digital Humanities, and Media Archeology, to name but a few. One can argue that these by now well-established programmatic ventures are symptomatic of a profound transformation taking place within the core epistemological apparatus of the Humanities: the fact that technology, rather than being a mere surrogate for the human body or the human mind, displaces and neutralises the sensible and the thinkable as it confronts our age-old ways of sense making.

In my dissertation I address and problematise the question of sense in a "computational" era and look at the various epistemological ruptures, sutures, and recursions that have effected profound transformations in how we conceive of ways of making sense. The focal point and main object of my dissertation is computation and its various historical moments, articulations, and artefacts. Throughout eight chapters, I address questions of language (*logos*) and its sense-effects in relation to mathematics, logic, software execution, and debugging, as well as works of modern literature. I read and problematise this *logos* through three types of idioms: *onto-logic*—asking the question "what is"—, *logo-logic*—asking the question "according to which rules"—, and finally *deonto-logic*—asking the question "what ought to be."

In *Chapter 0: Turing's Machines* I discuss the transformation of the meaning of the term "computing" from a pre-Turing era to Alan Turing's proposed formalisation of the term in 1936 with his concept of the Turing machine. Whereas computing, as a practice, was historically performed by human *computors* before WWII, and thus akin to the notion of calculating or reckoning, after the invention of the Turing machine, an understanding of computing as mere symbol manipulation became *de rigueur*; Turing's treatise radically demonstrated that any function which can be calculated by a human being can be computed by a Turing machine. I situate Turing's work within the field of metamathematics alongside the work of mathematicians David Hilbert (*Entscheidungsproblem*) and Alonzo Church

(effective calculability) and discuss how Turing machines put forth novel, albeit problematic, notions of "finite in-finite"—that is, a computation that never stops.

In *Chapter 1: Wittgenstein's Rules* I first address philosopher Ludwig Wittgenstein's enigmatic (and very short) comment on Turing's machines. Moving from a conception of a human machine (i.e., a computor doing calculations) to the one of an automatic machine, I ask: Who or what computes in the Turing machine? With Wittgenstein's writings on the foundation of mathematics I mark a difference between *logic mechanism* and *logic machine*, whereby the former relates to physical operations and the latter to these operation's finality or logical and mathematical sense-effect. I argue that the inexorability of the machine (i.e., the fact it it demands strict results) comes not from the logic mechanism but from the logic machine, which does not allow deviations. I then identify three different currents of thought that can explain such an inexorability, which I propose to categorise as: "Platonic," "anthropo," and "deonto-logic." In this chapter I look at the "Platonic" worldview, which metaphysically naturalises or transcendentalises logic and/or mathematics. I then look at how Wittgenstein rejects such onto-logic positions and refocuses the question on logo-logic grounds—that is, on mathematical rules and their following.

In *Chapter 2: Inhuman Obligations*, I continue my inquiry into Wittgenstein's philosophical investigations, and look at how—by denying that the rules of logic and mathematics can be deduced from reality and that therefore they are not strictly meant to represent reality—questions pertaining to the "internal" consistency of mathematics surface. I ask: Is it possible to devise a "pure" mathematical language where connections between its concepts can be defined and calibrated a priori? I discuss how Wittgenstein rebuffs such an idea by hinging the questions of the coherence and consistency of mathematics on the notion of "use." For Wittgenstein, mathematical statements and rules have no inherent meaning if they are not mobilised as part of a certain usage: statements and rules acquire a meaning through synthetic use and not the reverse. The aforementioned "anthropo" category is drawn to a close with a brief discussion on conventionalism—the idea that we humans set the rules: we invent languages and laws that fit (or not) certain problems we encounter. I conclude that while such conventionalism might reject the metaphysical posturing of the "Platonic" category, it nonetheless reifies the idea that humanly invented language can be directly deduced from empirical reality. I hence look elsewhere to grasp the logo-logic of Wittgenstein's rules and rule following, and turn to philosopher Georg von Wright's deontic logic. With what I call "deonto-logic," the questions of prescription and obligations, which are implied in the notions of rule and use, are put to the fore. In turn, a direct link can be drawn between rules of logic, mathematics, and the notion of law, opening up *logos* to questions of *nomos*.

In *Chapter 3: On Commands and Executions*, I offer a brief study of the notion of programming language from early developments in the 1940s to the advent of contemporary programming language paradigms in the late 1950s. I look at the various constructions that

were instrumental in supporting notions of programming language, including the devising of early software libraries, compilers, and interpreters. I argue that the advent of such meta-linguistic constructs, from the early 1960s onwards, had the effect of stratifying computing into software (symbolic) and hardware (material) practices. I argue that this ultimately worked to erase the peculiarities of hardware altogether by rendering all machines "virtual," as code written in a given programming language gained the ability to produce the same symbolic results and effects when executed on different machine architectures across various sites. I offer a critique of this type of "universalisation" of programming language by drawing connections between the theoretical concept of the law and the one of code or software. I offer a reframing of the static software/hardware divide into a dynamic and deonto-logic command-execution couple that I read as a constative-performative one.

In *Chapter 4: Céline's Literary/War Machine*, I turn to the literary work of French novelist Louis Ferdinand Céline and analyse his short novel *Casse-Pipe* [Cannon Fodder] in which he writes about the violent absurdity of the command-execution complex of the army. The aim of this chapter is to analyse Céline's use of language to express the violence of the army's command-execution complex and draw a link between the positions I developed in chapter 3. I argue that Céline's peculiar language can be conceived of as what philosophers Gilles Deleuze and Félix Guattari call a "minor music" in that it harbours a grammar of disequilibrium: syntax mangles sentences and words, creating nothing more than wefts of syntactically marked intensities, producing different rhythms at different speeds. I argue that Céline's language does not need to be endowed with a meaning that requires deciphering in order to make sense, but rather needs to be traversed at various speeds so that the intensities the text puts forth can be sensed. In considering Céline's critique of realist literature (i.e., Émile Zola's naturalist school) alongside Friedrich Nietzsche's notion of sense as effect, I argue that *Casse-Pipe* and Céline's language work within a *derealist* register of sense making.

In *Chapter 5: Logic of Rules or Rules of Logic?* I look at the act of writing a computer program and draw parallels with the act of writing mathematical proofs. I first discuss computer scientist Marvin Minsky's claim that a computer programmer does not necessarily need, or is even able, to know—in advance—how her/his program is to perform when executed. I recast Minsky's claim that the written rules of a program might not be so rigid after all into a problematic concerning determinability versus predictability and argue that these do not correspond. I show that this non-correspondence between determinability and predictability can be better framed as questions pertaining to philosopher Immanuel Kant's *analytic* versus *synthetic* distinction. With logician Jean-Yves Girard's philosophical work on logic and proof theory, I show that the constative-performative couple of chapter 3 can be embedded in a more general analytic-synthetic coupling so that the act of writing a computer program (or of writing a mathematical proof) can be understood as an act of construction that necessarily needs to be trialled and tested. Following Girard, I argue that programs are synthetic formats setting certain deonto-logic rules that ought to be followed

through trials and errors so that the questions they ask make sense. I argue that computational sense stems from this critical interplay between construction and deconstruction (usage) and cannot be given a priori from the formulation of the program alone (analytical definition). In other words, a program that does not execute is non-sensical as it does not have any sense in and of itself; it is only through its execution that any sense can become—execution conditions sense and not the other way around.

In *Chapter 6: Logical Realism and its Shadow: Constructivism*, I address the arguments of the main figures who opposed Kant's view that mathematics is a synthetic pursuit and who proposed that logic be reified as the analytical socle upon which all of mathematics is to be founded. I discuss logician Gottlob Frege's logic and I consider logicians Alfred North Whitehead and Bertrand Russell's colossal *Principa Mathematica*, a treatise set to unify and rewrite all domains of mathematics under the sole auspice of logic alone. I argue that at the core of what is commonly called the *logicism* worldview is an untenable stance towards the interrelated notions of *Sinn* (sense) and *Bedeutung* (reference). I show that because logicism reifies the analytic to the detriment of the synthetic, it can only concretise language's capacity of referencing and thus necessarily folds questions of sense into questions of signification alone. In turn, I look at the work of Dutch mathematician Luitzen Egbertus Jan Brouwer who formulated a radical critique of logicism's worldview by rejecting some of the core pillars of "classical" logic (e.g., the law of the excluded middle). What Brouwer's constructivism concurred to was to reform logic and mathematics, proposing a new type of proof theory giving rise to the so-called Brouwer-Heyting-Kolmogorov interpretation. I show how this interpretation fuelled remarkable developments in proof theory research from the mid-1930s onwards, leading to theories that formally link proof theory with computer programming. I argue that in rejecting logicism's notion of sense-as-reference constructivism is able to (re)institute a notion of sense that attends to the existential modalities of terms and signs that compose logical and mathematical proofs and propositions. I conclude the chapter by proposing to fold works of literature and mathematics that refuse the lure of comprehending sense as a signification into what I call "derealism," since these works short circuit the concept of realist signs by proposing signs and sign machines that are non-signifying yet productive of sense.

In *Chapter 7: On Rewriting Machines: Blank Signs & Types*, I look at formal signs and inquire how they are mobilised in mathematics, starting from David Hilbert's semiotic theories of mathematics. I analyse how Hilbert's signs are divorced from any *a priori* signification and are merely posited as syntactical markers or statements upholding a certain sense or directive to be followed. Formal signs, or *blank signs* as I call them, acquire meaning only when mobilised, *a posteriori*, as part of a certain construction. In other words, to give meaning to blank signs is to construct something with them: both meaning and construction are part of the same simultaneous gesture. I contrast the notion of blank signs to the asignifying signs presented in chapter 3 and show that since blank signs are formal they do

not address or point to the senses, they have no essential affective contexts or registers. With this consideration, I turn to type theory and type-token theory and look at how types, having no initial significative import, can be identified and constructed out of mere recurrence. That considered, I ask: What type of recurrence does the blank sign as a type require in order to gain an identity? Approaching this question I consider philosopher Quentin Meillassoux's treatment of the subject and examine his notions of repetition, iteration, and reiteration. While concurring with Meillassoux's analysis that a type does not follow a repetition, which is based on a perceptual synthesis of time, I reject his concepts of iteration and reiteration, suggesting a non-constructive notion of infinity, because they sabotage the very possibility of constructing types I argue. I thus turn to Derrida's notions of iteration to unyoke a kind of recurrence that can, at once, identify and differentiate signs. Based on this iteration, I propose the concept of *rewriting* to account for the type of recurrence found in formal systems and argue that computation, based on type (i.e., data type), can be framed as a rewriting apparatus that works with blank signs that have no a priori definitions nor meaning—that is, to compute involves the rewriting of blank signs.

As a conclusion to this dissertation, I discuss the politics underpinning the various conceptual and practical approaches to the study of computation I presented in the previous chapters. I argue that the constructivist and deonto-logic perspectives I offered are more prone and suited to address the *acts* of doing logic, mathematics, and computing compared to the classical onto-logic worldview that reifies a perspective of symbol manipulation that forecloses central questions of action and prescriptions. I then discuss the problems of commenting on the executory force of the deonto-logic, looking at how discursive commenting neutralises the power of action that is necessarily prompted by a prescription. I argue that the distance the discursive comment produces and the descriptive mode in which it is articulated is what neutralises the action's force. I thus seek to identify methods and politics that can frame a certain collapse of this neutralising distance of the theoretical comment. With this in mind, I look at ways of conceiving of theory in light of this collapsing of distance, proposing the notion of *debugging* to account for this collapse. Practically speaking, the impetus of debugging computer programs and/or circuits is precisely to approach and attend to the program's or circuit's execution (i.e., action) in a close manner. I argue that such a debugging mode of reading demands a transversal approach to theory that allows just enough situated distance to allow for the object or process to speak (of) itself—that is, reading with the object and not reading into it.

Samenvatting

To Execute, Rewrite, and Debug: Over de constructie en deconstructie van computatie

De huidige discoursen over digitale media kenmerken zich door een fascinatie voor de linguïstische, numerieke, algoritmische, sociale en materiële aspecten van technologie. Hieruit zijn in de afgelopen jaren verschillende nieuwe onderzoek domeinen voortgekomen, waaronder Digital Methods, Software Studies, Digital Humanities en Media Archeology, om er enkele te noemen. Men kan stellen dat deze inmiddels gevestigde programmatische ondernemingen symptomatisch zijn voor een diepgaande epistemologische transformatie die plaatsvindt binnen de Geesteswetenschappen: in plaats van slechts dienst te doen als surrogaat voor het menselijk lichaam of de menselijke geest, beïnvloedt en neutraliseert technologie het pragmatische en het denkbare, en confronteert technologie op die manier onze eeuwenoude tradities wat betreft zingeving.

In mijn proefschrift onderzoek ik de zingevingsvraag in een "computationeel" tijdperk en kijk ik naar verschillende epistemologische breuklijnen, verbindingen en recursie die diepgaande transformaties hebben teweeggebracht in de manier waarop we betekenis geven. Het brandpunt en hoofdonderwerp van mijn proefschrift is computatie en de daarmee samenhangende historische schakelmomenten, articulaties en artefacten. In acht hoofdstukken behandel ik vragen over taal (*logos*) en diens zinseffecten in relatie tot de mathematica, de logica, software-uitvoering en het debuggen, alsook het verband met werken uit de moderne literatuur. Ik lees en problematiseer deze logos door middel van drie soorten idiomen: *onto-logisch*—betreffende de vraag "wat is"—, *logo-logisch*—betreffende de vraag "volgens welke regels"—, en tenslotte *deonto-logisch*—betreffende de vraag "wat zou moeten zijn."

In *Hoofdstuk 0: Turing's Machines* bespreek ik de transformatie van de betekenis van de term "computing" van een pre-Turing tijdperk naar Alan Turing's voorgestelde formalisering van de term in 1936 met zijn concept van de "Turingmachine". Terwijl computatie voor de Tweede Wereldoorlog werd uitgevoerd door menselijke rekenaars (zogeheten *computors*, in het Engels), en dus historisch verwant is aan een zekere praktijk van berekenen of rekenen, werd na de uitvinding van de Turingmachine de computatie als louter symboolmanipulatie de standaard. Turing's verhandeling toonde op radicale wijze aan dat elke functie die door een mens kan worden berekend, kan worden berekend door een Turingmachine. Ik situeer Turing's werk binnen het veld van de metamathematiek naast het

werk van de wiskundigen David Hilbert (*Entscheidungsproblem*) en Alonzo Church (effectieve berekenbaarheid) en bespreek hoe Turingmachines nieuwe, zij het problematische noties van "eindig in eindigheid" naar voren brengen – dat wil zeggen, een berekening die nooit ophoudt.

In *Hoofdstuk 1: Wittgenstein's Regels* ga ik eerst in op de raadselachtige (en zeer summiere) commentaar van filosoof Ludwig Wittgenstein op de machines van Turing. Vertrekkende van het idee van een menselijke machine (d.w.z. een *computor* of rekenaar die berekeningen maakt) naar die van een automatische machine, stel ik de vraag: Wie of wat rekent er in de Turingmachine? Aan de hand van Wittgenstein's werk over het fundament van de wiskunde markeer ik een verschil tussen het *logische mechanisme* en de *logische machine*, waarbij de eerste betrekking heeft op fysieke bewerkingen en de tweede op de finaliteit, ofwel het logische en wiskundige betekeniseffect van deze bewerkingen. Ik beargumenteer dat de onverbiddelijkheid van de machine (d.w.z. het feit dat deze strikte resultaten eist) niet voortkomt uit het logische mechanisme maar uit de logische machine, die geen afwijkingen toelaat. Ik identificeer vervolgens drie verschillende stromingen die zo'n onverbiddelijkheid kunnen verklaren, die ik categoriseer als zijnde: "Platonisch," "anthropo," en "deonto-logisch." In dit hoofdstuk kijk ik naar het "Platonische" wereldbeeld, dat metafysisch de logica en/of de wiskunde naturaliseert of transcendent maakt. Vervolgens bekijk ik hoe Wittgenstein dergelijke onto-logische standpunten verwerpt en de vraag naar logo-logische gronden heroriënteert - dat wil zeggen, naar wiskundige regels en hun navolging.

In *Hoofdstuk 2: Niet-menselijke Verplichtingen*, verdiep ik mijn onderzoek naar Wittgenstein's filosofische studies, en kijk ik naar hoe, door te ontkennen dat de regels van de logica en de wiskunde kunnen worden afgeleid uit de werkelijkheid en dat ze daarom niet strikt bedoeld zijn om de werkelijkheid te representeren, bepaalde vragen met betrekking tot de "interne" consistentie van de wiskunde opkomen. Zo stel ik de vraag: is het mogelijk om een "zuivere" wiskundige taal te bedenken waarin de verbanden tussen de begrippen a priori kunnen worden gedefinieerd en gekalibreerd? Ik bespreek de manier waarop Wittgenstein zo'n idee afwijst door vragen over de samenhang en consistentie van de wiskunde te koppelen aan het begrip "gebruik". Voor Wittgenstein hebben wiskundige uitspraken en regels geen inherente betekenis indien ze niet worden gemobiliseerd als onderdeel van een specifiek gebruik: uitspraken en regels krijgen betekenis door synthetisch gebruik, en niet omgekeerd. De eerdergenoemde categorie "anthropo" wordt afgesloten met een korte discussie over het conventionalisme, i.e. het idee dat wij mensen de regels bepalen: we verzinnen talen en wetten die passen (of niet) bij bepaalde problemen die we tegenkomen. Ik concludeer dat dit conventionalisme weliswaar de metafysische houding van de "platonische" categorie afwijst, maar dat het niettemin het idee herbevestigt dat menselijk uitgevonden taal direct kan worden afgeleid uit de empirische werkelijkheid. Ik kijk daarom elders om de logo-logica van Wittgensteins regels en opvolging te begrijpen, en richt me tot de deontische logica van de filosoof Georg von Wright. Met wat ik "deonto-logica" noem, breng ik de noties van

voorschrift en verplichtingen, die in de begrippen regel en gebruik geïmpliceerd zijn, naar de voorgrond. Op deze manier wordt een direct verband gelegd tussen de regels van de logica, de wiskunde en de notie van het recht, waardoor *logos* wordt opengesteld voor vragen met betrekking tot *nomos*.

In *Hoofdstuk 3: Over Bevelen en Executies*, bied ik een korte studie aan van de programmeertaal vanaf de vroege ontwikkelingen in de jaren veertig tot de komst van de hedendaagse paradigma's aan het eind van de jaren vijftig van de vorige eeuw. Ik kijk naar de verschillende constructies die een ondersteunende rol hebben gespeeld bij de ontwikkeling van de programmeertaal, waaronder het ontwerpen van vroege softwarebibliotheken, compilers en tolken. Ik beargumenteer dat de komst van dergelijke meta-linguïstische constructies er vanaf het begin van de jaren zestig voor zorgde dat computergebruik werd gelaagd in software- (symbolische) en hardware- (materiële) praktijken. Ik stel dat dit uiteindelijk heeft geleid tot het uitwissen van de eigenaardigheden van de hardware doordat het alle machines "virtueel" heeft gemaakt: een code die in een bepaalde programmeertaal is geschreven, kan dezelfde symbolische resultaten en effecten produceren, ook wanneer deze binnen verschillende machinearchitecturen op verschillende sites wordt uitgevoerd. Ik bekritiseer dit soort "universalisering" van de programmeertaal door verbanden te leggen tussen het theoretische concept van de wet, en dat van de code of software. Ik herbedenk de statische software/hardware dichotomie binnen een dynamisch en deonto-logisch commando-uitvoerings tweetal, dat ik benader als een constatief-performatief koppel.

In *Hoofdstuk 4: Céline's Literaire/Oorlogsmachine* verdiep ik me in het literaire werk van de Franse schrijver Louis Ferdinand Céline en analyseer ik zijn korte roman *Casse-Pipe*, waarin hij schrijft over de gewelddadige absurditeit van het commando-uitvoeringscomplex van het leger. Het doel van dit hoofdstuk is om Céline's taalgebruik omtrent het geweld van het commando-executie complex van het leger te analyseren, en een verband te leggen met de stellingen die ik in hoofdstuk 3 heb ontwikkeld. Ik stel dat Céline's eigenaardige taal kan worden opgevat als wat de filosofen Gilles Deleuze en Félix Guattari een "*musique mineure"* of minor muziek noemen, in die zin dat er een grammatica van onevenwichtigheid in zit: de syntaxis vermengt zinnen en woorden, waardoor er niets anders ontstaat dan inslagen van syntactisch gemarkeerde intensiteiten, die verschillende ritmes met verschillende snelheden voortbrengen. Ik beargumenteer dat de taal van Céline geen betekenis moet krijgen die ontcijferd moet worden om zinvol te zijn, maar dat deze met verschillende snelheden moet worden doorkruist, zodat de intensiteit van de tekst overkomt. Tenslotte plaats ik Céline's kritiek op de realistische literatuur, met name de naturalistische school van Émile Zola, naast Friedrich Nietzsche's notie van zingeving als effect, en beargumenteer ik dat *Casse-Pipe* en Céline's taalgebruik in het algemeen binnen een *derealistisch* register van betekenisgeving opereren.

In *Hoofdstuk 5: Logica van Regels of Regels van Logica?* behandel ik het schrijven van een computerprogramma en trek daarbij parallellen met het schrijven van mathematische

bewijzen. Ik bespreek eerst de bewering van computerwetenschapper Marvin Minsky, dat een computerprogrammeur niet noodzakelijkerwijs op voorhand hoeft te weten, of zelfs maar in staat is te weten, hoe diens programma moet presteren – zolang het wordt uitgevoerd. Ik herinterpreteer Minsky's bewering dat de geschreven regels van een programma misschien toch niet zo rigide zijn tot een probleemstelling met betrekking tot bepaalbaarheid versus voorspelbaarheid, en argumenteer daarbij dat deze niet overeenkomen. Ik toon aan dat deze non-correspondentie tussen bepaalbaarheid en voorspelbaarheid beter gekaderd kan worden binnen het analytische versus synthetische onderscheid, ontwikkeld door de filosoof Immanuel Kant. In combinatie met het filosofische werk van logicus Jean-Yves Girard over logica en bewijsvoering, laat ik zien dat het constatief-performatieve tweetal uit hoofdstuk 3 ingebed kan worden in een meer algemene analytisch-synthetische koppeling, zodat de handeling van het schrijven van een computerprogramma (of van het schrijven van een mathematisch bewijs) kan worden begrepen als een constructieve handeling die noodzakelijkerwijs moet worden uitgeprobeerd en getest. In navolging van Girard stel ik dat programma's synthetische patronen zijn die bepaalde deonto-logische regels opstellen die door middel van trial en error moeten worden opgevolgd om zinvol te zijn. Ik beargumenteer dat computationele betekenis voortkomt uit de kritische wisselwerking tussen constructie en deconstructie (het gebruik), en daarom niet a priori kan worden aangebracht in de formulering van het programma alleen (de analytische definitie). Met andere woorden, een programma dat niet wordt uitgevoerd is zinloos omdat het op zichzelf geen betekenis draagt; alleen door de uitvoering ontstaat zingeving – executie is de voorwaarde voor betekenis, en niet andersom.

In *Hoofdstuk 6: Logisch Realisme en Diens Schaduw: Constructivisme*, ga ik in op de argumenten van de hoofdspelers die zich verzetten tegen Kant's opvatting dat de wiskunde een synthetisch streven is en die stellen dat de logica moet worden gereorganiseerd als de analytische sokkel waarop alle wiskunde moet worden gegrondvest. Ik bespreek de logica van logicus Gottlob Frege en ik beschouw daarbij ook de logici Alfred North Whitehead en Bertrand Russell's kolossale *Principa Mathematica*, een verhandeling die bedoeld is om alle domeinen van de wiskunde te verenigen en te herschrijven onder de enige auspiciën van de logica alleen. Ik beargumenteer dat de kern van wat gewoonlijk het logica-wereldbeeld wordt genoemd een onhoudbare houding is ten opzichte van de onderling samenhangende begrippen *Sinn* (betekenis) en *Bedeutung* (referentie). Ik laat zien dat, omdat het *logicisme* het analytische ten nadele van het synthetische herstelt, het slechts de capaciteit van de taal om te refereren kan concretiseren, en dus noodzakelijkerwijs vragen met betrekking tot betekenis automatisch beperkt tot vragen met betrekking tot significatie. Ik kijk vervolgens naar het werk van de Nederlandse wiskundige Luitzen Egbertus Jan Brouwer die een radicale kritiek op het wereldbeeld van de logica formuleerde door enkele basisprincipes van de 'klassieke' logica (bijvoorbeeld het principe van het uitgesloten midden) te verwerpen. Het constructivisme van Brouwer onderbouwde de hervorming van de logica en de wiskunde, waarbij hij een nieuw type van bewijstheorie voorstelde dat aanleiding gaf tot de zogenaamde

Brouwer-Heyting-Kolmogorov-interpretatie. Ik laat zien hoe deze interpretatie opmerkelijke ontwikkelingen in het onderzoek naar de bewijstheorie vanaf het midden van de jaren dertig heeft gevoed, die op hun beurt hebben geleid tot theorieën die de bewijstheorie formeel verbinden met computerprogrammering. Ik beargumenteer dat, door het verwerpen van de betekenis-als-referentie notie van het logicisme, het constructivisme de zintuiglijke waarneming kan (her)inrichten die zich richt op de existentiële modaliteiten van termen en tekens die logische en mathematische bewijzen en stellingen samenstellen. Ik sluit het hoofdstuk af met het voorstel om literaire en mathematische werken die de verleiding van de betekenis als significatie verwerpen, te omvatten in wat ik het "derealisme" noem, aangezien deze werken het concept van realistische tekens kortsluiten door tekens en tekenmachines voor te stellen die niet betekenisvol zijn maar wel productief qua zingeving.

In *Hoofdstuk 7: Over Herschrijfmachines: Lege Tekens & Types*, kijk ik naar formele tekens en onderzoek ik hoe deze worden gemobiliseerd in de wiskunde, waarbij ik uitga van David Hilbert's semiotische theorieën over de wiskunde. Ik analyseer hoe Hilbert's tekens gescheiden zijn van enige a priori betekenisgeving en slechts gepositioneerd worden als syntactische markeringen of verklaringen die een bepaalde betekenis of richtlijn inhouden die moet worden opgevolgd. Formele tekens, of lege tekens zoals ik ze noem, krijgen pas betekenis als ze a posteriori, als onderdeel van een bepaalde constructie, worden gemobiliseerd. Met andere woorden, het geven van betekenis aan lege tekens houdt een constructie met deze tekens in: zowel betekenis als constructie maken tegelijkertijd deel uit van dezelfde geste. Ik stel het begrip "lege tekens" tegenover de in hoofdstuk 3 gepresenteerde niet-betekenis gevende tekens en laat zien dat, aangezien lege tekens formeel zijn, ze niet op de zintuigen gericht zijn en dus ook geen wezenlijke affectieve contexten of registers hebben. Met deze overweging wend ik me tot de typetheorie en de type-token theorie en kijk ik hoe types, die in eerste instantie geen significante betekenis hebben, kunnen worden geïdentificeerd en geconstrueerd louter op basis van herhaling. Dit beschouwende, stel ik de vraag: Wat voor soort herhaling heeft het lege teken als type nodig om een identiteit te krijgen? Bij het beantwoorden van deze vraag neem ik de behandeling van het onderwerp door de filosoof Quentin Meillassoux in overweging, en onderzoek ik zijn noties van herhaling, iteratie en re-iteratie. Hoewel ik het eens ben met Meillassoux' analyse dat een type geen herhaling volgt, hetgeen gebaseerd is op een perceptuele synthese van de tijd, verwerp ik zijn concepten van iteratie en re-iteratie, die een niet-constructieve notie van oneindigheid suggereren, omdat deze concepten de constructie van de types die ik behandel onmogelijk maakt. Ik wend me dan tot Jacques Derrida's noties van iteratie om een soort herhaling te onderkennen die tegelijkertijd tekens kan identificeren en onderscheiden. Op basis van deze iteratie, stel ik het concept van *herschrijven* voor dat rekening houdt met het type herhaling dat in formele systemen aanwezig is, en beargumenteer ik dat de computatie, gebaseerd op het type (i.e. het data type), kan worden omkaderd als een herschrijfinstrument dat werkt met

lege tekens die geen a priori definities of betekenis hebben - dat wil zeggen, computatie houdt in het herschrijven van blanco tekens.

Als conclusie van dit proefschrift bespreek ik de politiek die ten grondslag ligt aan de verschillende conceptuele en praktische benaderingen van de studie van computatie, die ik in de voorgaande hoofdstukken heb gepresenteerd. Ik noem daarbij dat de constructivistische en deonto-logische perspectieven die ik voorstel meer geneigd en geschikt zijn om het *beoefenen* van logica, mathematica en computatie aan de orde te stellen, in vergelijking met het klassieke onto-logische wereldbeeld dat een perspectief van symboolmanipulatie herbevestigt en daarbij centrale vragen van actie en voorschrift uitsluit. Vervolgens bespreek ik de problemen rond het becommentariëren van de executieve kracht van de deonto-logica, waarbij ik kijk hoe discursieve commentaar de kracht van actie, die noodzakelijkerwijs wordt ingegeven door een voorschrift, neutraliseert. Ik argumenteer dat de afstand die de discursieve commentaar produceert en de beschrijvende modus waarin deze is gearticuleerd, de kracht van de actie neutraliseert. Ik identificeer vervolgens methoden en politieke kaders voor zo'n ineenstorting van de neutraliserende afstand voor theoretische kritiek. Met dit in het achterhoofd kijk ik naar manieren om theorie te benaderen in het licht van deze ineenstorting van afstand, waarbij ik de notie van *debugging* als verklaring voorstel. Praktisch gezien is de impuls van het debuggen van computerprogramma's en/of circuits juist om de uitvoering van het programma of circuit (i.e. de actie) van dichtbij te benaderen en op te volgen. Ik stel dat zo'n *debugging*-leesmodus een transversale benadering van theorie vereist die net genoeg gesitueerde afstand toestaat om het object of het proces (over) zichzelf te laten spreken - dat wil zeggen, lezen met het object en niet het inlezen ervan.