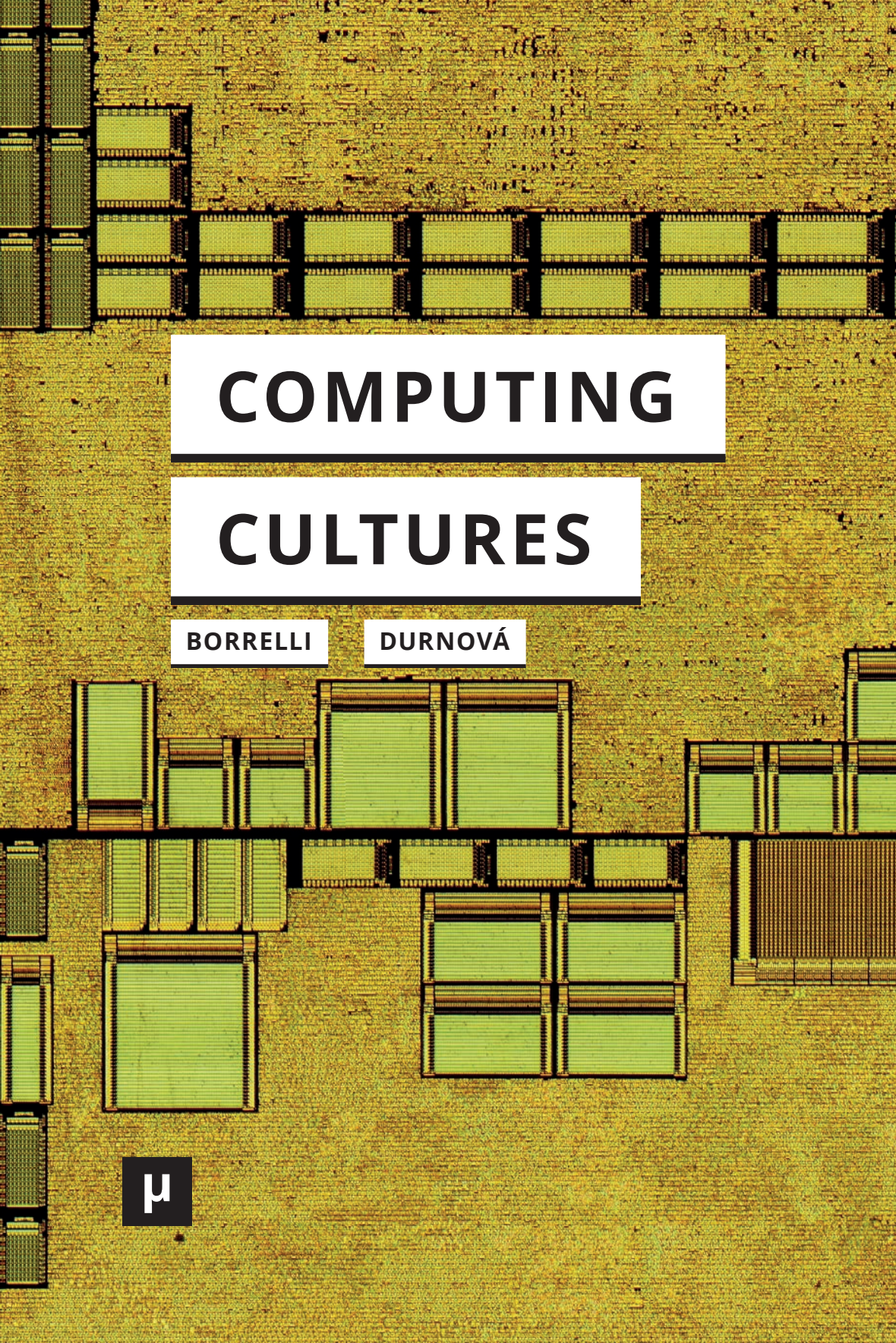


The background of the entire cover is a high-resolution, close-up photograph of a microchip. The chip's surface is a complex grid of gold-colored circuitry, with various rectangular blocks of different sizes and shapes, some of which are highlighted in a bright yellow-green color. The overall texture is highly detailed and technical.

COMPUTING CULTURES

BORRELLI

DURNOVÁ

μ

Contents

Contents	7
Introduction	10
[1] Priestley: Synthetic Machines and Practical Languages	30
[2] Léon: Practical and Theoretical Objectives of Early Machine Translation in the.	64
[3] Borrelli: Big Machines for Big Science	86
[4] Mori: The Influence of Organization and Methods on Early Business Computing	124
[5] Durand-Richard: The Division of Mental Labor in Computing Practices	150
[6] Lejeune: Mainframe Computer or Programmable Pocket Calculator?	196
[7] Bredella: Computing Practices, Data-Based Design, and Knowledge Cultures...	230
[8] Astarte: Tense and Temporality	258
Authors	284

Computing Cultures

Computing Cultures: Knowledges and Practices (1940–1990)

Edited by

Arianna Borrelli and Helena Durnová



meson press

Bibliographical Information of the German National Library

The German National Library lists this publication in the Deutsche Nationalbibliografie (German National Bibliography); detailed bibliographic information is available online at dnb.d-nb.de.

Published in 2025 by meson press, Lüneburg, Germany
www.meson.press

Design concept: Torsten Köchlin, Silke Krieg
Cover photo: Microscopic image of a 3Dfx Voodoo 3 3000 by
FritzchensFritz@Flickr, <https://www.flickr.com/photos/130561288@No4/27522083907/>, CCo
Copy editing: Selena Class

The print edition of this book is printed by Lightning Source,
Milton Keynes, United Kingdom.

ISBN (Print): 978-3-95796-273-7
ISBN (PDF): 978-3-95796-274-4
DOI: 10.14619/2737

The digital edition of this publication can be downloaded freely at:
www.meson.press

This publication is licensed under a CC BY-SA 4.0 (Creative Commons Attribution ShareAlike 4.0 International) license. To view a copy of this license, visit: <http://creativecommons.org/licenses/by-sa/4.0/>.



Copyright and permissions for the reuse of all external content marked as such (e.g. images, photos, diagrams etc.) differ from the above. They are excluded from the CC BY-SA 4.0 license, and it may be necessary to obtain further usage permission from the respective rights holders for their reuse.

meson press, Salzstr. 1, 21335 Lüneburg, Germany
info@meson.press

Contents

Introduction 7

Arianna Borrelli and Helena Durnová

[1] Synthetic Machines and Practical Languages: Masking the Computer in the 1950s 27

Mark Priestley

[2] Practical and Theoretical Objectives of Early Machine Translation in the 1950–60s 61

Jacqueline Léon

[3] Big Machines for Big Science: The Beginnings of Scientific Computing at CERN 83

Arianna Borrelli

[4] The Influence of Organization and Methods on Early Business Computing 121

Elisabetta Mori

[5] The Division of Mental Labor in Computing Practices: Presuppositions, Advances, Biases 147

Marie-José Durand-Richard

[6] Mainframe Computer or Programmable Pocket Calculator? Calculation Tools and Practices of Computing in Medieval History (1960s–1980s) 193

Edgar Lejeune

**[7] Computing Practices, Data-Based Design, and Knowledge Cultures
During the Post-War Period 227**

Nathalie Bredella

[8] Tense and Temporality: Computing and the Logic of Time 255

Troy Kaighin Astarte

Authors 281

Introduction

**Arianna Borrelli and
Helena Durnová**

1. The History and Historiography of Computing: A Few Remarks

During the late 1940s, electronic, digital computers built by state-funded academics started to be used in a few fields of science and technology in the USA and Europe.¹ By the early 1960s computers had become successful commercial products with increasingly broad and varied user communities. During the 1970s computers diversified into a range of machines satisfying different needs: from high-performance, expensive mainframes (service included) for affluent clients from big business, administration and defense, to more compact, customizable machines that had a lower performance and came with little or no servicing, but were affordable and desirable for academic researchers and smaller businesses. In those years software ceased being included in hardware purchases as a free add-on and became a commercial product in its own right, marking the beginning of a new and rapidly expanding market. From the late 1970s onward, thanks to the availability of cheap electronic components, personal computers became increasingly popular despite their very modest performance, at first only among hobbyists with a passion for electronics, but later also among the general public. By the late 1990s a significant portion of US and European households owned a computer and so the emerging World Wide Web had a sufficiently large user base to persist.

¹ For an overview of the history of electronic, digital computers mainly in the Western world, see Campbell-Kelly et al. (2023); Gugerli (2022); and Haigh and Ceruzzi (2021). For the developments in the Soviet Block see Trogemann, Nitussov, and Ernst (2001).

The contributions collected in this volume address a number of chosen topics from this broad landscape, and in doing so also demonstrate some of the different, often interdisciplinary approaches to the history of computing which have been developed since its inception in the 1970s. The earliest texts in the field were first-hand accounts by actors who had participated in the narrated developments, such as those written by the mathematician Herman Goldstine (1972) and the physicist Nicholas Metropolis (1980), or were written on behalf of firms that had become major players in computer manufacturing, most notably IBM, for which in 1971 Charles and Ray Eames designed the exhibition *A Computer Perspective* and prepared the relevant catalog (Eames and Eames 1973). These works focused on specific, often rather technical, developments close to the interests of the authors and were quite influential in the computing community, creating historical lineages for the new field of expertise. They also inspired some of the earliest museum exhibitions on the history of computers, which were often designed by practitioners, as for example the section *Informatik und Automatik* at the Deutsches Museum in Munich, inaugurated in 1988 and shaped by the computer scientist Friedrich Bauer (Hashagen 2008, 121–22). Indeed, some of these early accounts are today better known among computer experts than the works by professional historians that started appearing from the 1980s onward. Among the latter contributions the works by Michael Mahoney stand out for their historiographical relevance: Mahoney, originally a historian of mathematics, turned to the history of computing during the 1980s and shaped later historiography both with his own research and through a number of methodological papers which underscored the necessity of setting the development of computing within the broader context of the history of science, technology, and society (Mahoney 1988, 2011). The relevance of the broader socioeconomic picture for understanding the development of computing was underscored early on also by the French psychologist and anthropologist Philippe Breton in his *Histoire de l'informatique* (1987). The 1990s saw the publication of the first monographs offering comprehensive and contextualized accounts of the history of computing. Historian of science William Aspray and business historian Martin Campbell-Kelly teamed up to reconstruct the history of the computer as “information machine” by embedding it in a US economical and political context reaching back to the 19th century (Campbell-Kelly and Aspray 1996). Historian of technology and museum curator Paul Ceruzzi instead let his *History of Modern Computing* begin in the years immediately after the Second World War and reconstructed the broader technological and scientific developments which enabled the spread of electronic, digital computers (Ceruzzi 1998). Another seminal work also appeared: Paul Edward’s *The Closed World: Computers and the Politics of Discourse in Cold War America* (1996), highlighting the role of the computer not only as a technological tool, but also as an ideological template for US politics in the Cold War era.

Since then the field has broadened in all directions, as witnessed by the fact that both Campbell-Kelly and Aspray's and Ceruzzi's overviews have been repeatedly updated to keep up with the rapid development of computing practices and computing historiography, also enlisting the help of further historians: Gerardo Con Diaz, Nathan Ensmenger, Thomas Haigh, Honghong Tinn, and Jeffrey Yost (Campbell-Kelly et al. 2014, 2023; Campbell-Kelly and Aspray 2004; Haigh and Ceruzzi 2021). It is not possible to offer here an overview of the breadth and variety that the field has achieved today, but there are some features of its development which we wish to discuss, based on recent historiographical reflections on the field, as they provide the motivation for the present volume (Abbate and Dick 2022a; Cortada 2019a, b; De Mol and Bullynck 2018; Gugerli and Zetti 2019). A main result of these studies is to show how research on different aspects of computing has been shaped by the sociocultural contexts and disciplinary interests of those writing it, and so eventually gave rise to a set of largely independent historiographical traditions. Of course this is true also for the historiography of other fields of science and technology, as historians can never escape the situated perspectives from which they reconstruct and therefore interpret past events, particularly those regarding subjects that have seen such recent and rapid development as computing. Even taking this into account, though, the historiography of computing seems to be characterized by an extraordinary degree of fragmentation, so that offering an overview of it is a challenge in itself. One difficulty is that computing straddles the borders between science, technology, logic and mathematics, belonging to all and none; mainstream historians from those fields have so far either devoted limited attention to developments in computing or focused only on the historical aspects nearer to their area of expertise. For example, developments in computer science and programming have mainly been explored by historians with backgrounds in mathematics, computing or the natural sciences, as well as by philosophers of science (De Mol and Primiero 2018; Haigh, Priestley, and Rope 2016; Mahoney 2011; Nofre 2023; Priestley 2018; Varenne 2019), while research more focused on artefacts is usually located in specialized museums (Deutsches Museum, Computer History Museum), where the issue of software preservation was also addressed early on (Hashagen, Keil-Slawik, and Norberg 2002; Rojas and Hashagen 2000; Shustek 2006). Hardware and software manufacturing, computerization in industry and administration, and patent issues have been the focus for historians of technology and business (Campbell-Kelly 2004; Con Diaz 2019; Cortada 2012), while artists and architects have explored the interplay of computing, art and design in society (Bredella 2022; Martin 2003; Sack 2019), and scholars from science and technology studies took a critical look at the implications of computing practices in today's cultures and societies (Mullaney et al. 2021; Roberge and Castelle 2021; Vertesi et al. 2019).

All these studies have offered outstanding contributions to exploring the past of today's computing practices, yet the historiography of computing remains fragmented and its results have found limited reception among the broader historian community. This situation is probably one reason why historians of computing have rarely contributed to discussions on the premises and implications of digitalization in science, technology and society, while reflections on this topic by philosophers, sociologists and scholars from cultural and media studies have become increasingly prominent and would benefit from an in-depth historical contextualization. Only recently a number of works demonstrated the productivity of a broader historical perspective for critically approaching today's computing practices. Among them are essay collections revisiting "the early digital" from an interdisciplinary perspective (Haigh 2019), exploring the diverse interplay of digital tools and society (Wichum and Zetti 2022), discussing the tension between the abstract and the concrete in the history of computing (Abbate and Dick 2022b), historicizing the oppositions between a rule-bound way of working characteristic of (computer) code and a more creative craft usually associated with the human mind and body (Evans and Johns 2023), or highlighting the role of ideal visions and imaginaries in shaping the history of computing (van Lente 2022). Artificial intelligence (AI), too, has finally received attention on the part of historians of science, who have contextualized and questioned its origin narrative, highlighting the variety of situated developments and overarching connections to broader themes like management and colonialism and complementing the traditional US-centred narratives with European perspectives (Ali et al. 2023; Piel and Seising 2023). As is usually the case with outstanding historical work, it is not possible to summarize their results in a few sentences, as their strength lies in variety and richness of detail, bringing to light a network of expected and unexpected connections which provide new questions and new material for philosophical, cultural, and sociological analyses.

The perception of this so far underexploited potential of the history of computing for broadening current discussions on the so-called Digital Turn is a main motivation for the present volume, which stands in the tradition of a history of computing trying to bridge the gap between the technoscientific and the socioeconomic approaches to the subject, while also taking into account its epistemological implications. As noted above, this gap is not unique to the history of computing, though it is particularly evident in it. Writing the history of the development of science after the Second World War poses great challenges, since an understanding of the subject is often necessary to reconstruct the institutional and social history, while at the same time it is also important for historians to distance themselves from, and critically reflect on, the implicit assumptions, values and conceptual templates shared by the community they are studying, be they computer experts, physicists or biologists. Looking at computing as a set of contextualized

practices is a most effective approach that does not leave out the practitioners themselves and their technical endeavor, but takes into account also their institutional, social, political, cultural, and economic environment. This approach has been actively promoted by Liesbeth De Mol and Giuseppe Primiero, who organized a series of interdisciplinary events on the history and philosophy of computing, starting with a conference in Ghent in 2011, in which one of the present editors, Helena Durnová, participated. The efforts by De Mol and Primiero led to the establishment of the DHST/DLMPST Interdivisional Commission of History and Philosophy of Computing (HaPoC), under whose sponsorship a symposium was organized at the International Conference of History of Science and Technology held in virtual mode in 2021. The papers in the symposium provided the starting point for the present volume and, in the symposium as in the volume, the aim has been to promote exchange both between different branches of history of computing and with the broader community of historians of science and technology. For this reason, no specific topic within that broad field was chosen as a guideline, as that might have limited possible contributions. Instead, the papers were put together by inviting authors with different disciplinary backgrounds and methodological approaches to contribute, asking them to make their research questions and (tentative) answers accessible and appealing to a non-specialistic audience, while maintaining the highest standards of historical reconstruction. All authors rose up to and successfully met this challenge and we are deeply thankful for the work they invested in the contributions that make up this volume. We believe that these texts will both present new results to expert scholars and allow readers from diverse disciplinary and methodological backgrounds to approach and appreciate subjects they are less—or not at all—familiar with. The similarities and differences between the case studies can provide material for the comparisons that are necessary to gain a deeper understanding of digitalization, and we hope that the contributions, with all their necessary limitations in themes and cultural frameworks, will prompt further initiatives in this direction.

2. Historicizing the Digital Turn

The investigations collected in the following pages range from *longue durée* studies questioning traditional assumptions about the nature and origin of computing practices to detailed reconstructions of situated developments in commercial computer manufacturing or scientific computing. These case studies provide a sample of a broad and varied landscape, essentially focusing on the years between 1950 and the early 1990s. This period is today often regarded as having laid the foundations for the increasingly rapid diffusion of computational methods in all fields. Indeed, those years saw the expansion of the computer industry, the beginning of stand-alone software developments,

programming languages and informatics, a variety of more or less successful attempts at deploying computing machines in new areas of public and private life and, in the later phase, the advent of personal computers and the internet. Yet the period deserves a study for its own sake, not just as the “origin” of later developments. In fact, a closer look at computing practices in those years shows that what looks like a linear success story of computational technologies from a distance actually comprises a diversity of developments due to very different, situated factors, and whose success was in no way as undisputed as often presented *a posteriori*. Becoming acquainted with this variety is not only historically significant but also essential for assessing more recent constellations.

In this sense, the contributions complement, and at times correct, narratives which often feature in discussion about present implications of computing, but appear problematic when looking more closely at the historical events. A prominent example is the idea that, during the second half of the 20th century, the diffusion of computing in an increasing number of areas gave rise to significant, possibly revolutionary, transformations. “Computer revolution,” “information age,” “digital turn” and, more recently, “AI revolution:” these and similar expressions have been used to refer both to techno-scientific developments and to cultural and social phenomena related to them. Philosophers, sociologists and scholars from media and cultural studies have often presented the computer as a universal machine whose appearance prompted radical changes in all areas of science, technology, and society, although this focus was already challenged by sociologist Susan Leigh Star in her introduction to the essay collection *The Cultures of Computing* (Denning and Tedre 2019; Duran and Eckhart 2013; Hartmann 1996; Kittler 1989; Pias 2011; Star 1995, 7; Turkle 2009; Winsberg 2010). Historians, while investigating these developments, have instead for the most part remained skeptical of revolutionary claims and rather noted the variety of computing practices, from both synchronic and diachronic perspectives, as well as the continuities between practices before and after the alleged revolution (Borrelli and Wellmann 2019; Campbell-Kelly et al. 2023; Cortada 2019a, 2019b; De Mol and Bullyinck 2022; Haigh and Ceruzzi 2021). In 2006 historian Jon Agar provocatively asked “What difference did computers make?,” suggesting that computers had less innovative impact than usually assumed and that, more often than not, their uses were deeply shaped by the cultural-historical contexts in which they were embedded (Agar 2006).

The tension between continuity and change in the history of science and technology has often been at the center of discussions both among historians and with philosophers, a paradigmatic example being the debate on whether a diversity of processes taking place at different times during the premodern and early modern period may or not be seen as adding up to a “scientific

revolution" (Shapin 1996). In our case, historians today generally agree that no single, linear history of computing can be written, and that one should actually rather speak of "histories of computing" in the plural, as suggested by Mahoney (2011). Even the claim that the diffusion of computing in commercial enterprises and administration was motivated by an increase in efficiency and productivity has been questioned: historians of economy now regard it as a "productivity paradox" that computing tools could spread in commercial fields despite usually not increasing, and at times decreasing, productivity (Cortada 2019a, 2019b). Overviews on the history of computing have variously dealt with the tension between continuity and change: Campbell-Kelly and Aspray characterize the computer as "the information machine," thus linking it with a notion that became prominent only after the Second World War, but at the same time they let the history of that machine begin in the 19th century (Campbell-Kelly and Aspray 1996, 2004; Campbell-Kelly et al. 2014, 2023). Tom Haigh and Paul Ceruzzi (2021) on the one hand describe computers as striving towards universality from the Second World War onwards, yet they present this development as a process of diversification in which the computer acted as a "universal solvent," slowly penetrating all areas of modern culture and society by adapting to them and becoming in turn a "data processing machine," a "scientific supertool," or a "personal plaything." David Gugerli (2022) makes instead the reverse suggestion, namely conceiving the history of computing as a narrative of "how the world got into the computer." All these approaches make clear how processes of digitalization are at the same time extremely varied and deeply interconnected, posing great challenges to historians of computing, of science and technology, and of society and culture.

Following Agar's suggestion, we believe that the emergence and diffusion of a broad range of computers, computing tools, their relevant users, and computer-related research and commercial initiatives can be best understood in terms of the emergence of a landscape of computing cultures that could share some features while remaining largely independent from each other. Beside Agar, other authors have underscored this cultural variety of computing practices. In her overview of the development of computer science in the USA, Stephanie Dick spoke of different "computing communities," each with its own vision of what the computer should be (Dick 2015, 60–1), and more recently the proceedings of the interdisciplinary HaPoC Conference held in Zurich in 2021 were devoted to *Computing Cultures: Historical and Philosophical Perspectives* (Gastaldi 2024). The contributions in the present volume offer in-depth historical examples from this landscape. Unlike Agar, though, we are not specifically interested in whether computers made a difference in one or the other historical context, but rather wish to reconstruct the situated interplay of computing practices, knowledges, and cultures. In this way, we understand "computing" in a broad sense, including not only hardware, software and computer science, but also the procedures and tools for deploying them in

specific fields, as well as the images of computing and computers characteristic of specific cultural-historical contexts. We investigate computing cultures focusing on mutually entangled knowledges and practices, approaching computing as a performance which cannot be fully separated from its environment: the humans enacting it, their often shifting goals, the cultural and sociopolitical contexts they inhabit, the material apparatuses needed to perform computations, and the formal and conceptual frameworks shaping the act of computing—from programming languages to smartphone design. In other words, one has to reconstruct the spectrum of situated knowledges and practices shaping and being shaped by emerging computing cultures.

3. The Contributions

The papers explore a broad range of historical contexts, but two themes run through the whole volume. The first one is the way in which the diffusion of computing practices from the 1950s onward was realized in specific cultural-historical constellations, like disciplines, research institutions, and commercial and technological enterprises. The results of these studies offer an outstanding example of the situatedness of early computing practices, while at the same time uncovering connections and similarities which deserve further investigation. The second theme is revisiting from new perspectives origins and origin stories of notions and disciplines today associated with computing, showing that what may appear today as straightforward was in fact much less obvious at the time. The two themes complement each other, demonstrating how the investigation of the past can help critically reassess the present.

The volume opens with Mark Priestley's original perspective on a classical topic of the early history of computing: the combined development during the 1940s and '50s of electronic digital computers and of ways of making them more accessible to users with little or no previous experience of programming. These events have been traditionally framed in terms of the emergence of (higher level) programming languages, and today it is difficult to discuss the subject avoiding language-inspired terminology. However, there is no reason to assume that this way of conceptualizing developing programming practices was already dominating the discourse at the time, and Priestley presents evidence of an alternative way of conceiving them, namely as the creation of "synthetic machines" that masked the actual machines to the users. Synthetic machines were constructed by means of subroutines, following an approach pioneered by the research group led by Maurice Wilkes at Cambridge University and further developed in close exchange with Charles Adams' team working on the Whirlwind computer at MIT. The subroutines were shaped by the needs of the targeted users, who could be students, administrative staff, mathematicians, or linguists. Conceiving these user-friendly programming environments as synthetic machines for a time peacefully coexisted with the

conceptualizations that eventually came to dominate, namely programming languages. Rediscovering this lost perspective brings us nearer to the historical actors, allowing us to appreciate the openness and novelty of their tasks. Yet the relevance of the paper goes well beyond a reconstruction of past ways of thinking, because it highlights the complex interdependence between the physical computer with its machine code on the one side and the synthetic machine with its specialized input notation on the other. This interdependence persists today, despite computing practices often being conceived in the simplistic terms of the hardware/software dichotomy. Indeed, arguably the interdependence and hybrid nature of the layers of computational tools have even increased, given the enhanced medial spectrum of human-computer interaction, which today includes touch screens, voice and pattern recognition, Global Positioning System (GPS) localization, and more. Reconstructing the emergence of these layers and the different ways they could and can be conceived is therefore of extreme interest for the current discussion on digitalization.

One of the synthetic machines/programming languages discussed by Priestley is COMIT, whose target users were linguists working in the 1950s at MIT on the development of machine translation, whose structure was shaped by the views of Noam Chomsky on language functioning. Chomsky was himself a member of the MIT project, which was in turn part of a broader initiative aimed at machine translation promoted by the US government between 1949 and 1966. This initiative is at the center of Jacqueline Léon's paper, which looks back at what is today often seen as the "origin story" of machine translation, albeit a rather unsuccessful one, as these early attempts did not bear practical fruits. From today's standpoint, it is indeed tempting to interpret the statements and practices of these historical actors as if they were pursuing the same goal as later scientists, but without achieving the envisioned results. However, in her analysis Léon shows how the visions and goals of these early attempts at developing computational methods for analyzing languages and translating texts were much more varied than would later be the case, ranging from the mechanization of dictionaries to a fully automated translation or a computer-based analysis of fundamental language structures. This broad spectrum reflected the diversity of the backgrounds of the scientists involved and the novelty of their approaches to exploring and assessing the potential of computers for language manipulation. In particular, Léon underscores how the work of these authors was shaped by the tensions between their (often overly optimistic) goals and the constraints of technical, political and economic natures with which they were confronted. From these tensions emerged various reflections and questions, some of which may seem today naïve or obsolete, while others remain of great relevance, such as the question of how much translation error should be regarded as generally acceptable. Therefore, it appears too narrow to regard these early efforts as defective

versions of later technological successes, and Léon suggests revisiting from this perspective also the 1966 report of the Automatic Language Processing Advisory Committee (ALPAC). This report advised the US government against further investment in machine translation for economical reasons, effectively ending research on the topic for the time being. However, as Léon notes, the text also highlighted how the efforts so far had produced new ideas on the relationship between language studies and computing that could be, and indeed were, productive in research both in linguistics and in programming. In fact, both the term and the discipline of computational linguistics may be seen as having their origin in the report. The papers by Priestley and Léon demonstrate how the diversity of the history of computing can and should be recovered not only by looking at so far understudied actors and contexts, but also by approaching from a new perspective historical constellations that have already been studied.

Other than the two former contributions, the paper by Arianna Borrelli is devoted to a subject which received surprisingly little attention on the part of historians of computing, namely the introduction of computers in physics research. More specifically, the paper offers a broadly contextualized reconstruction of the process which in 1955/56 led to the purchase of the first electronic digital computer at the research laboratory today known as the European Organization for Nuclear Research (CERN) and located at Geneva in Switzerland. Philosophers of science have been debating since the 1990s the way in which computing might have more or less radically changed science, and more specifically fundamental physics (Galison 1997; Hartmann 1996; Winsberg 2010). When assuming that such a revolutionary change took place, the diffusion of computers in that field can appear as a straightforward development whose motivation hardly needs to be questioned: computers were the right tool for the job, and scientists recognized it. Borrelli's paper instead delivers evidence of widespread lack of interest in computers among high-energy physicists at CERN around the middle of the 1950s. The acquisition of the first computer for the laboratory was the result of a rather isolated initiative in which scientists at the institute only slowly started taking interest. Borrelli presents a largely counterintuitive picture of the beginning of computing at CERN, in which the machine was not acquired in response to specific computational needs, but rather due to a vague impression among research leaders and science managers that computers could be made into new scientific instruments for high-energy physics. Yet the necessary condition for this development was that physicists—as opposed to engineers, mathematicians, or manufacturers—should take charge of the new machines and turn them into scientific tools. How did this impression spread among high-energy physicists, though, before computers became a familiar presence in their laboratories? Here we meet again Maurice Wilkes, who featured already in Priestley's paper. It appears that the Mathematical Laboratory led

by Wilkes in Cambridge in the early 1950s was a hub promoting exchange between computer developers and the growing community of potential computer users in scientific research, especially in the exact sciences. A closer look at these networks may shed further light on the processes of digitalization in the natural sciences.

In that period, Wilkes' group at Cambridge also played a quite different role: establishing from 1947 onward a partnership with the British catering company J. Lyons and Co. and supporting their plan of designing and using computers for business management, eventually branching into computer manufacturing, which was realized in 1954 by founding LEO Computers Limited. This, in many ways, surprising story is carefully pieced together on the basis of so far unexplored archival material in Elisabetta Mori's paper. The EDSAC, the electronic digital computer built at Cambridge in the late 1940s, was the starting point for building what would become the LEO I, an electronic digital machine designed for business applications that eventually evolved into a series of LEO II and III machines sold worldwide. As Mori explores in much detail, while the technical expertise for this project came from Cambridge University, the motivation and the vision behind it were clearly due to the Lyons management, which had very early on recognized the potential of computers as tools for efficient administration. In fact, it was the interest in innovative methods of office management that drove Lyons to explore how far the new computers could further enhance efficiency, and it was with this vision in mind that Lyons employees travelled to the USA in 1947 to learn more about the new machines and their possible uses in industrial management. The tradition of scientific management of Frederick Taylor and Lillian and Frank Gilbreth had been taken up and further developed at Lyons, and formal tools of management like flowcharts contributed to shaping the LEO approach to coding, merging with other methods developed by mathematicians and engineers. This case study shows how important it is to reconstruct the specific details of a given computing culture without prejudice. In a commercial joint venture of scientists and managers, one might have taken for granted that the former would take care of the technical side and the latter of financing and marketing, but in this case there was no such clear-cut division of labor. Mori's results confirm and expand the argument recently made by historians of AI that management practices played an important, so far underestimated, role in the development of computing methods (Ali et al. 2023).

The connection between the history of computing and developments in industry and management is also at the center of the contribution by Marie-José Durand-Richard. The paper takes as its starting point a claim often repeated in early histories of computing, namely that the structure of electronic, digital computers can be seen as the realization of the principle of division of labor which Charles Babbage promoted in the 19th century as an

ideal template for organizing not only industrial work, as Adam Smith had initially showed, but also politics and scientific activities such as mathematical computation. From this point of view modern computers have been presented as embodying the ideal which Babbage failed to realize with his planned mechanical computing machines, the Difference Engine and the Analytical Engine. In a fascinating *longue durée* study spanning one and a half centuries Durand-Richard challenges this narrative by reconstructing and contextualizing both Babbage's visions and the later development of machines for numerical computation, from the first mechanical office machines and analog computers of the late 19th century to the early electronic digital machines of the years after the Second World War. She brings to light the diverse, historically and culturally situated connections between the various calculating tools and the political, economical and social contexts they were embedded in, with particular attention to organizational and managerial structures. From this background she convincingly argues that modern computers do not result only from the principle of division of labor as envisaged by Babbage, nor can Babbage's Engines be regarded as anticipating them. Babbage's machines have to be understood as situated expressions of the political, economical and managerial cultures of their time. These were very different from those which, a century later, would foster the emergence of computing as a technological, economic and managerial phenomenon of the period after the Second World War. It was not simply a lack of money or political support that forced Babbage to abandon his project, but rather the lack of interactions between actors, and of necessary social, industrial and technological premises, which Babbage could hardly conceive with his top-down view of science.

While Durand-Richard's paper demonstrates the many-layered relationships between computing devices and their historical contexts as they unfolded over more than a century, the contribution by Edgar Lejeune addresses this issue through a case study involving a smaller circle of actors over a shorter time span, bringing to light an equally large variety. The study focuses on a subject which has only recently started to attract attention: the history of Digital Humanities. What is or is not a computer in a given context? How do computing machines relate to the cultures in which they come to be embedded? Lejeune investigates these crucial questions in a comparative analysis of computing practices among medievalists from the late 1960s to the early 1980s. This research combines a detailed analysis of published and archival sources and oral history interviews with a methodological approach from cultural studies. The paper demonstrates the variety of computing tools and methods within a single community, showing how technical choices are entangled with issues of disciplinary methodology and science policy. Most interestingly, these results highlight the role played by a largely underestimated type of electronic tools that usually do not even qualify as "computers," namely the programmable pocket calculators that came on the

market in the early 1970s. Lejeune compares two historical research programs: the large international cooperation led by David Herlihy and Christiane Klapisch-Zuber from the late 1960s onward with the aim of analyzing the *catasto* (land registry) of Florence from the year 1427, and the studies conducted by French historian Alain Guerreau in the 1970s and 80s. While the former cooperation made use of large mainframe computers and storage facilities in the form of magnetic tapes, the latter's technical means of analysis was a programmable pocket calculator with which he derived semi-qualitative results. As Lejeune shows, the specific choices of computing methods were shaped not only by the research goals, but also by different views on the aims of the discipline, its optimal organization, and the envisioned role of computers in that context. At the same time, the specific features of the computational tools used could determine research questions and methods and contribute to the emergence of a situated research culture of historians in which programming either mainframe computers or pocket calculators was an essential component. Lejeune thus shows that the emergence of the Digital Humanities was neither a straightforward continuation of earlier practices with new means nor a methodological revolution brought about by digitalization, but rather a multi-layered process whose study must take into account the diversity of computing devices, research goals, available infrastructures, and disciplinary visions. Computers did not simply diffuse among historians: specific communities of historians appropriated computers if, when and how they saw them as fitting tools for the job. Here a parallel could be drawn with Borrelli's argument that in the later 1950s some high-energy physicists began to look at computers as potentially useful devices to be appropriated and transformed into fitting instruments for their own discipline, independently of whether a factual need for computing power existed before their introduction.

The connection between visions of computing and disciplinary innovation also features prominently in Nathalie Bredella's discussion of the project developed in the years after the Second World War at national and international level by the Greek architect Constantinos Doxiadis, a project for which he coined the name "*ekistics*," a science exploring with empirical and theoretical means the structures of human settlements. According to Doxiadis, the principles of *ekistics* could be deployed at all scales, from small villages and ancient Greek *poleis* to large-scale urbanization projects, and they also addressed general questions on the interaction between humans and the environment that were later framed in terms of ecology. In the 1960s Doxiadis was among the first architects not only to envision practices of what would later be called computer-assisted design, but also try and actually implement this vision in cooperation with both computer manufacturers and policy-makers. Despite these industrial and political connections, Doxiadis' computing practices maintained a visionary character which was essential

to their appeal as tools for large-scale, international reconstruction programs such as those funded by the Rockefeller Foundation. Thanks to the image of the computer as a tool for management and innovation, architects could present themselves as mediators at a geopolitical level, channeling and operationalizing the plans of technocratic institutions. Moreover, architects were in charge not only of planning and building, but also of assessing the situation and potential problems in specific urban settlements through qualitative and quantitative surveys of various kinds, whose results could be computationally analyzed. In this vision, on the one hand the computer became a container for a whole settlement, while on the other hand the city could be regarded as a working machine comprising both infrastructure and inhabitants. Bredella's analysis reconstructs the ideals and activities of Doxiadis and his collaborators, highlighting both the innovative visions and the internal tensions of the projects, thus providing fascinating insights into the early interplay of computing and architecture, as well as a further perspective on the multi-layered connections between computing and management.

While in Bredella's paper the computer appears in a double role as ideal template for conceiving the world and practical tool to manage it, in Troy Astarte's contribution it features rather as an abstract, universal device to be programmed in accordance with coherent principles that can be stated in terms of logical structures. The computing culture dealt with in Astarte's paper is that of computer scientists, focusing on the entangled development of concurrent programming and temporal logic between the 1960s and early 1990s. The founders of computer science usually came from a background in mathematics or formal logic, and the connection to those disciplines remained strong even after the new field achieved its full autonomy during the 1960s. From this background, it is tempting to conceive the relationship between developing computing practices and the emergence of computer science in terms of either a progressive formalization of the practices or an application of formal structures to programming. Yet the picture is more complex, challenging the distinction between principles and applications. Astarte offers a highly original, insightful example of the tensions between the formal and the practical, and of the diversity of interests and goals that could shape the development of what *a posteriori* can be seen as a theoretical construct situated at the intersection of logic and computer science. The paper focuses on the emergence of the field today known as temporal logic, which deals with possible logical formalization of the role of time sequence in establishing the logical truth or falsity of a statement. The question of how to deal with utterances that can be true or false depending on when they are pronounced had been addressed since antiquity, but it was only during the 20th century that frameworks to systematically explore the potential roles of time in logic emerged. From the 1960s onward, many of these developments were linked to the necessity of predicting and controlling the behavior of two or more

programs running in parallel on the same machine and potentially leading to errors by trying to access the same resources at the same time. These kinds of questions in computer science go under the name of concurrency. Ensuring that no conflict situation arose was a problem that some authors assumed could be productively tackled with the help of logical analysis of the concurrent programs, an analysis in which consideration of what became called temporal logic played a central role. Yet in the process of addressing these practical issues, the structure and workings of concurrent programs could in turn become abstract templates to formulate and explore new questions of logic. Astarte carefully reconstructs the different paths which led various historical actors to take an interest in and further develop formal reflections on the logic of time, appropriating and building upon each other's work, and the cooperations and conflicts that ensued. The paper demonstrates how the emerging research field was neither an extension of age-old questions of logic nor a set of practice-oriented rules for programming, but rather a new knowledge area whose coming to be was linked both to developing programming practices and to the establishment of the culture of computer science. Here a parallel can be drawn with the emergence of practices of Digital Humanities as reconstructed by Lejeune. In that case, too, the historical analysis shows how a new knowledge field came to be neither as a consequence of the introduction of new computational tools nor as a continuation of old traditions with new means, but rather thanks to the multi-layered interplay of disciplinary cultures and computer-aided practices within the field of medieval studies.

The papers included in this volume are a first, limited attempt at bringing together results from different branches of the historiography of computing. The aim is to showcase diversity and interest in the landscape, and to highlight how historiography might at present profit not only from the further development of the different branches, but also from increased cross-fertilization between them and closer cooperation with the broader community of historians of science and technology. Focusing on the years between 1940 and 1990, the studies reconstruct a number of past computing cultures which, although they may now have disappeared, nonetheless played a role in shaping those that came after them by facilitating the development of new computer-related practices and ways of knowing. Understanding these past cultures, their interconnections, and their broader historical context is therefore essential to deal critically with the present of computing, and especially to be able to contextualize and fairly reassess the visions, dreams, fears and tensions characterizing digital practices in today's knowledge societies.

Our deepest gratitude goes to Liesbeth De Mol, without whose initiative and inspiration not only this volume, but also the Commission for History and Philosophy of Computing (HaPoC) would not exist. We further wish to thank all the participants in the HaPoC Symposium at the ICHST 2021, including of course also those who could not contribute to the volume. For the institutional and financial support for the present publication we are indebted to the DFG-funded Institute for Advanced Study on Media Cultures of Computer Simulation (MECS) at Leuphana University Lüneburg (DFG grant KFOR 1927). Arianna Borrelli also gratefully acknowledges the financial support and intellectual inspiration experienced during her stay at the Käte-Hamburger-Kolleg "Cultures of Research" at the RWTH Aachen in 2022–23.

References

- Abbate, Janet, and Stephanie Dick. 2022a. "Thinking with Computers." In *Abstractions and Embodiments: New Histories of Computing and Society. Studies in Computing and Culture*, edited by Janet Abbate and Stephanie Dick, 1–19. Baltimore: Johns Hopkins University Press.
- Abbate, Janet, and Stephanie Dick, eds. 2022b. *Abstractions and Embodiments: New Histories of Computing and Society. Studies in Computing and Culture*. Baltimore: Johns Hopkins University Press.
- Agar, Jon. 2006. "What Difference Did Computers Make?" *Social Studies of Science* 36, no. 6: 869–907.
- Ali, Syed Mustafa, Stephanie Dick, Sarah Dillon, Matthew L. Jones, Jonnie Penn, and Richard Staley, eds. 2023. "Histories of Artificial Intelligence: A Genealogy of Power." *British Journal for History of Science (BJHS) Themes* 8: 1–18.
- Borrelli, Arianna, and Janina Wellmann, eds. 2019. "Computer Simulations Then and Now: An Introduction and Historical Reassessment." *NTM* 27: 407–17.
- Bredella, Nathalie. 2022. *The Architectural Imagination at the Digital Turn*. Abingdon: Routledge.
- Breton, Philippe. 1987. *Une histoire de l'informatique*. Paris: La Découverte.
- Campbell-Kelly, Martin. 2004. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Cambridge, MA: MIT Press.
- Campbell-Kelly, Martin, and William Aspray. 1996. *Computer: A History of the Information Machine*. New York: Basic Books.
- . 2004. *Computer: A History of the Information Machine*. Second edition. Boulder: Westview Press.
- Campbell-Kelly, Martin, William Aspray, Nathan Ensmenger, and Jeffrey R. Yost. 2014. *Computer: A History of the Information Machine*. Third edition. Boulder: Westview Press.
- Campbell-Kelly, Martin, William Aspray, Jeffrey R. Yost, Honghong Tinn, et al. 2023. *Computer: A History of the Information Machine*. Fourth edition. New York: Routledge.
- Ceruzzi, Paul E. 1998. *A History of Modern Computing*. Cambridge, MA: MIT Press.
- Con Díaz, Gerardo. 2019. *Software Rights: How Patent Law Transformed Software Development in America*. New Haven: Yale University Press.
- Cortada, James W. 2012. *The Digital Flood: Diffusion of Information Technology across the United States, Europe, and Asia*. Oxford: Oxford University Press.
- . 2019a. "Rethinking Why and How Organizations Acquire Information Technologies, Part 1." *IEEE Annals of the History of Computing* 41: 48–60.
- . 2019b. "Rethinking Why and How Organizations Acquire Information Technologies, Part 2." *IEEE Annals of the History of Computing* 41: 61–70.
- De Mol, Liesbeth, and Maarten Bullynck. 2018. "Making the History of Computing: The History of Computing in the History of Technology and the History of Mathematics." *Revue de Synthèse* 139: 361–80.
- . 2022. "What's in a Name? Origins, Transpositions and Transformations of the Triptych Algorithm-Code-Program." In *Abstractions and Embodiments: New Histories of Computing and Society*, edited by Janet Abbate and Stephanie Dick, 146–68. Baltimore: Johns Hopkins University Press.

- De Mol, Liesbeth, and Giuseppe Primiero, eds. 2018. *Reflections on Programming Systems: Historical and Philosophical Aspects*. Cham: Springer.
- Denning, Peter J., and Matti Tedre. 2019. *Computational Thinking*. Cambridge, MA: MIT Press.
- Dick, Stephanie. 2015. "Computer Science." In *A Companion to the History of American Science*, edited by Georgina M. Montgomery and Mark A. Largent, 55–68. Hoboken: Wiley-Blackwell.
- Duran, Juan M., and Eckhart Arnold, eds. 2013. *Computer Simulations and the Changing Face of Scientific Experimentation*. Newcastle upon Tyne: Cambridge Scholars Publishing.
- Eames, Charles, and Ray Eames. 1973. *A Computer Perspective*. Cambridge, MA: Harvard University Press.
- Edwards, Paul N. 1996. *The Closed World: Computers and the Politics of Discourse in Cold War America*. Cambridge, MA: MIT Press.
- Evans, James, and Adrian Johns, eds. 2023. "Beyond Craft and Code: Human and Algorithmic Cultures, Past and Present." *Osiris* 38. <https://doi.org/10.1086/725077>.
- Galison, Peter. 1997. *Image and Logic: A Material Culture of Microphysics*. Chicago: University of Chicago Press.
- Gastaldi, Juan Luis, ed. 2024. "Computing Cultures: Historical and Philosophical Perspectives." Special Issue *Minds and Machines* 34. <https://link.springer.com/collections/dgcfbghba>.
- Goldstine, Herman H. 1972. *The Computer from Pascal to von Neumann*. Princeton: Princeton University Press.
- Gugerli, David. 2022. *How the World Got into the Computer: The Emergence of Digital Reality*. Zurich: Chronos.
- Gugerli, David, and Daniela Zetti. 2019. "Computergeschichte als Irritationsquelle." In *Provokationen der Technikgeschichte*, edited by Martina Heßler and Heike Weber, 193–228. Paderborn: Verlag Ferdinand Schöningh.
- Haigh, Thomas, ed. 2019. *Exploring the Early Digital*. Cham: Springer.
- Haigh, Thomas, and Paul E. Ceruzzi. 2021. *A New History of Modern Computing*. Cambridge, MA: MIT Press.
- Haigh, Thomas, Mark Priestley, and Crispin Rope. 2016. *ENIAC in Action: Making and Remaking the Modern Computer*. Cambridge, MA: MIT Press.
- Hartmann, Stephan. 1996. "The World as a Process." In *Modelling and Simulation in the Social Sciences from the Philosophy of Science Point of View*, edited by Rainer Hegselmann, Ulrich Mueller, and Klaus G. Troitzsch, 77–100. Dordrecht: Springer. https://doi.org/10.1007/978-94-015-8686-3_5.
- Hashagen, Ulf. 2008. "Der Computer als Ausstellungsobjekt: Eine kurze Geschichte von Ausstellungen zur Geschichte des Computers." In *Technik – Faszination und Bildung*, edited by Arnold Vogt, 112–30. München: Müller-Straten.
- Hashagen, Ulf, Reinhard Keil-Slawik, and Arthur L. Norberg, eds. 2002. *History of Computing: Software Issues*. Berlin: Springer.
- Kittler, Friedrich A. 1989. "Fiktion und Simulation." In *Philosophien der neuen Technologie*, edited by ARS ELECTRONICA, 57–80. Berlin: Merve.
- Mahoney, Michael S. 1988. "The History of Computing in the History of Technology." *Annals of the History of Computing* 10, no. 2: 113–25. <https://doi.org/10.1109/MAHC.1988.10011>.
- . 2011. *Histories of Computing*, edited by Thomas Haigh. Cambridge, MA: Harvard University Press.
- Martin, Reinhold. 2003. *The Organizational Complex: Architecture, Media, and Corporate Space*. Cambridge, MA: MIT Press.
- Metropolis, Nicholas C., ed. 1980. *A History of Computing in the Twentieth Century: A Collection of Essays*. New York: Academy Press.
- Mullaney, Thomas S., Safiya Umoja Noble, Benjamin Peters, Mar Hicks et al., eds. 2021. *Your Computer Is on Fire*. Cambridge, MA: MIT Press.
- Nofre, David. 2023. "'Content Is Meaningless, and Structure Is All-Important': Defining the Nature of Computer Science in the Age of High Modernism, c. 1950–c. 1965." *IEEE Annals of the History of Computing* 45: 29–42. <https://doi.org/10.1109/MAHC.2023.3266359>.

- Pias, Claus. 2011. "On the Epistemology of Computer Simulation." *Zeitschrift für Medien- und Kulturforschung* 2 no. 1: 29–54. <https://doi.org/10.28937/1000107521>.
- Piel, Helen, and Rudolf Seising, eds. 2023. "Perspectives on Artificial Intelligence in Europe." *IEEE Annals of the History of Computing* 45, no. 3: 6–79.
- Priestley, Mark. 2018. *Routines of Substitution: John von Neumann's Work on Software Development, 1945–1948*. Cham: Springer.
- Roberge, Jonathan, and Michael Castelle, eds. 2021. *The Cultural Life of Machine Learning: An Incursion into Critical AI Studies*. Cham: Springer.
- Rojas, Raúl, and Ulf Hashagen, eds. 2000. *The First Computers: History and Architectures*. Cambridge, MA: MIT Press.
- Sack, Warren. 2019. *The Software Arts*. Cambridge, MA: MIT Press.
- Shapin, Steven. 1996. *The Scientific Revolution*. Chicago: University of Chicago Press.
- Shustek, Len. 2006. "What Should We Collect to Preserve the History of Software?" *IEEE Annals of the History of Computing* 28 no. 4: 112–111.
- Star, Susan Leigh. 1995. "Introduction." In *The Cultures of Computing*, edited by Susan Leigh Star, 1–28. Oxford: Blackwell Publishers.
- Trogermann, Georg, Alexander Y. Nitussov, and Wolfgang Ernst, eds. 2001. *Computing in Russia: The History of Computer Devices and Information Technology Revealed*. Braunschweig: Vieweg.
- Turkle, Sherry, ed. 2009. *Simulation and Its Discontents*. Cambridge, MA: MIT Press.
- van Lente, Dick, ed. 2022. *Prophets of Computing: Visions of Society Transformed by Computing*. New York: Association for Computing Machinery.
- Varenne, Franck. 2019. *From Models to Simulations*. Abingdon: Routledge.
- Vertesi, Janet, David Ribes, Carl DiSalvo, Laura Forlano, et al., eds. 2019. *DigitalSTS: A Field Guide for Science & Technology Studies*. Princeton: Princeton University Press.
- Wichum, Ricky, and Daniela Zetti, eds. 2022. *Zur Geschichte des digitalen Zeitalters*. Wiesbaden: Springer.
- Winsberg, Eric B. 2010. *Science in the Age of Computer Simulation*. Chicago: University of Chicago Press.

PROGRAMMING LANGUAGES

VIRTUAL MACHINES

AUTOMATIC CODING

WHIRLWIND

LANING AND ZIERLER

[1]

Synthetic Machines and Practical Languages: Masking the Computer in the 1950s

Mark Priestley

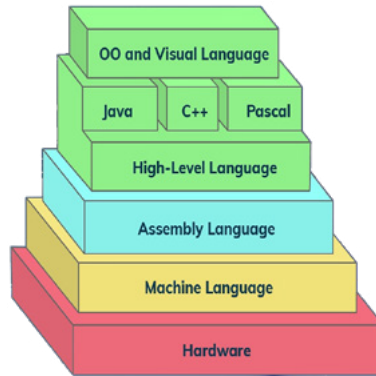
In the 1950s, programming was seen as an arcane practice and a likely bottleneck in the use of the new high-speed automatic computers. One proposal to solve this problem was to enable scientists and others to use computers directly in their work. To make this feasible, improved notations were designed to mediate between users and the machine. These notations are often understood purely linguistically, as ways to describe the domain-specific knowledge and techniques of a particular field in a machine-processible way. However, these improved notations also interpreted the computer for the non-specialized user. Using a new notation gave users access to a “virtual” or “synthetic” computational device often simpler or better tailored to a specific application area than the available physical machine. This chapter examines the synthetic machines of the 1950s, with a focus on those developed in connection with MIT’s Whirlwind

machine, and shows how programming languages bridged apparently incompatible areas of knowledge, allowing users to grasp some of the ideas underlying the new computers and simultaneously express at least some of their domain-specific knowledge and problems in a computational form. In conclusion, it is suggested that this history challenges the simple hardware/software dichotomy which continues to shape much thinking about the computer itself.

The automatic digital computer was, from the moment of its appearance in the 1940s as a novel tool for scientific calculation, understood as a linguistic agent, an interpretation clearly visible in the common description of programming as the preparation of problems in “the language the machine can understand” (Burks, Goldstine, and von Neumann 1946, preface). This attribution of agency placed computers in a long, if subaltern, tradition of thought about “active machinery” defined in opposition to the idea of “brute mechanism” relying solely on the transfer of momentum (Riskin 2016). Understood as language users, computers shared the imaginative space occupied by Karel Čapek’s robots, for whom a crucial boundary was the one across which orders were transmitted (Čapek 1920, 66). Even before it was a writing machine (Dick 2013), the computer was an obeying machine: Bell Labs mathematician George Stibitz defined a computer’s “native intelligence” as “its ability to understand and obey a large number of orders” (Stibitz 1943), while Alan Turing, commenting on the Automatic Computing Engine’s (ACE) versatility, suggested that “we should consider the machine as doing something quite simple, namely carrying out orders given to it in a standard form which it is able to understand” (Turing [1946]).

Language was initially seen as an organic part of a computer and programming as a communicative act between human and machinic agents, but throughout the 1950s, as computers became more widely available, this collaboration (or confrontation) was increasingly mediated by notations that were easier for humans to use and which the computer itself could translate into its “native language,” a practice known as “automatic coding.”¹ By the end of the

1 The metaphor of translation was widely used, despite the fact that translation between natural languages is nothing like the mechanical meaning-preserving transformations carried out by programming language compilers. There may be a connection here with the parallel project of machine translation, itself informed by cryptographic experience during the Second World War, for which see Léon in this volume.



[Fig. 1]: Typical contemporary representation of programming language hierarchies

decade, as the intermediate processing grew in sophistication, some of these notations had come to be described as “programming languages.” Expected benefits from the use of mediating notations included reductions in the cost, staffing, and time required for program preparation, greater modifiability and portability of programs, and increased versatility of computers.

Nofre, Priestley and Alberts (2014) describe this process as a transformation from technology to language, driven by the need for computer installations to preserve their investment in software by ensuring interoperability among suites of incompatible machines. They thematize independence from the machine, noting in particular how universalizing projects such as Algol aimed to produce notations that were abstract and formal like the languages of mathematics and logic. In this narrative, notation evolves away from the machine, from machine code through symbolic assembly languages to higher-level programming languages, and the resulting hierarchical vision is still a widespread way of thinking about programming languages (see Figure 1). Language types are thought to represent specific “levels of abstraction” and the process that allows one to move down the hierarchy and eventually use the invisible machine lurking at the bottom of the stack is understood as one of translation.

Many researchers in the 1950s shared this vision. But automatic coding was also understood as a process that worked on the machines themselves, a technique for transforming an awkward and hard-to-use computer into one that was more accessible. These improved computers had no distinct material existence but possessed different logical characteristics from the underlying physical machines. Such spectral devices are now often referred to as “virtual machines,” but to avoid confusion we will adopt a phrase coined at the time and refer to the computers conjured up by automatic coding systems as “synthetic machines” (Backus 1959). Massachusetts Institute of Technology (MIT) programmer Frank Heart made early use of this notion of synthesis,

describing automatic coding as “a class of methods which synthesize programmed digital computers ‘within’ actual digital computers” (Heart 1954).

In a recent discussion of the origins of computer science, Nofre has drawn attention to a related phenomenon, describing how “the conceptualization of the activity of computing became disentangled from the computer itself, a conceptual shift that was closely intertwined with *a process of dematerialization of the very notion of the computer*” (Nofre 2023, emphasis added). He focuses on the work of university staff anxious to escape the “commercial capture” of the computer and to establish a research role in computing distinct from the research and development carried out by commercial organizations such as IBM (International Business Machines). However, a differently motivated group inhabited the space between the universities and commercial computer developers, namely computer programmers and users concerned above all with making easy and economic use of the new machines in real-world applications. For this group, “dematerializing” the computer was not a way of making the physical machine disappear but rather of improving access to it.

This chapter traces the development of the discourse on synthetic machines as it evolved throughout the 1950s in parallel with linguistic accounts of programming. An early symposium paper described three automatic coding systems developed for MIT’s Whirlwind computer (Adams and Laning 1954): these make convenient type specimens for a preliminary classification of synthetic machines and are described in sections 2 to 4. Sections 5 and 6 then consider how the idea of synthetic machines persisted, albeit sometimes in an attenuated and implicit way, as attention increasingly turned to the development of programming languages. But we begin by considering the basic socio-economic driver of this activity, namely the need to make programming easier.

1. Users as Programmers

As early as 1946 it was recognized that programming was a potential bottleneck in the use of electronic digital computers. In a lecture that summer, John Mauchly, principal designer of the ENIAC (Electronic Numerical Integrator and Computer), explained that a basic economic issue posed by the new machines was the incompatibility between extremely high speeds of computation and the time-consuming task of program preparation (Mauchly 1946a). Mauchly further noted that coding was the only aspect of computer use where substantial automation appeared not to be possible, but he hoped that well-designed instruction codes would lead to appreciable reductions in cost. A similar concern was expressed three years later at the initial meeting of the UK government’s advisory committee on computers, when computation expert Douglas Hartree commented that “at least in the immediate future, it would

be difficult to find enough programme material to keep [100] machines in operation" (DSIR 1949).

One strategy to address this problem was to turn users into programmers. Rather than relying exclusively on a specialized cadre of machine operators, scientists and engineers would write their own programs with advice and guidance being provided by experts where necessary. This approach was pioneered by Maurice Wilkes's group at the University of Cambridge's Mathematical Laboratory, who organized a summer school on program design in September 1950, only a year after their new electronic computer, the EDSAC (Electronic Delay Storage Automatic Calculator), had become operational. The prospectus for the course described the "large contribution" that computers could make "to the solution of problems in mathematics, physics, engineering and other subjects," and while acknowledging the specialized nature of programming, stated that "the subject will be treated throughout from the point of view of the mathematical user" (UML 1950). Historian Doron Swade writes that the EDSAC approach created "the user as a new class of practitioner" (Swade 2011), and one former user described the laboratory as a "disparate" yet "close-knit" community of "mathematicians, crystallographers, molecular biologists, theoretical chemists, physicists, astronomers and statisticians" developing programs related to their individual research (Barron 2011).

These developments had a profound influence on the Whirlwind group at MIT, similarly situated in an elite techno-scientific institution and surrounded by scientists and engineers eager to get their hands on any new computational technology. Although Whirlwind was largely dedicated to specific military projects, a portion of its time was allocated without charge for general use by MIT staff and students (Adams 1952), and in the summer of 1948 Alan Perlis and John Carr were employed as research assistants to work alongside Charles Adams developing coding technique. Carr was subsequently awarded a Fulbright scholarship to study in France and on his journey home spent some time in Cambridge, returning to MIT filled with enthusiasm for what he had learned there (Wilkes 1985, 177). As Perlis recalled:

[Carr] showed us something called a symbolic code that he had seen at Cambridge ... Charlie Adams was the head of our programming group and we turned to each other [and said] "That's the way to do business!" and from that followed immediately invitations to the British to send us your rich, your intelligent, and so forth. (Perlis 1983, min. 17:47)

Wilkes himself visited MIT in August 1950 and discussed the EDSAC approach with Adams. Immediately after the meeting, Adams reported that Wilkes's "philosophy is both a confirmation and an extension of ideas" independently developed at MIT, noting in particular that experience with EDSAC confirmed the feasibility of "allowing users to learn coding by actually doing their own

problems" (WW M-1092). He listed three key technical points—fully automatic loading of programs from tape with no machine set-up required, the use of standard conventions and a subroutine library, and time-effective methods of trouble-shooting—that were also emphasized by Wilkes in a report on the Cambridge summer school (Wilkes 1950). While in the USA Wilkes gave two talks on "Operational experience on the EDSAC" at a meeting of the Association for Computing Machinery (ACM) in September. Adams described the meeting as "exceptionally stimulating" and noted that he had made personal contacts that had "encouraged work [at MIT] towards a practical, convenient, fully automatic means for handling coding details and computer set-up" (WW M-1096). Rounding off these early interactions, the Whirlwind group soon received a copy of the long and detailed internal report on EDSAC programming that was the basis for the famous textbook published the following year by Wilkes, David Wheeler, and Stanley Gill (WW M-1119).

As Whirlwind became increasingly reliable, Adams' applications group began to think how it could best be used as a research tool. They aimed:

to show the feasibility of using a fast, general-purpose computer to solve many relatively small-scale scientific and engineering problems (where a single problem may require only a minute or two of machine time once a day or once a month). (WW SR-25, 28)

To ensure productive use of the machine's time under such a regime, "efficient methods of using the machine and efficient methods of locating troubles in programs" were essential, and Adams' group adopted and adapted the EDSAC framework, developing input and conversion programs that performed a similar job to EDSAC's Initial Orders, an extensive subroutine library, and a variety of tools for debugging. The end result of this work was the "Comprehensive System of Service Routines" (CSSR), which became available to Whirlwind users towards the end of 1952 (WW E-516).

These developments were bolstered by ongoing personal contacts. David Wheeler, creator of EDSAC's Initial Orders, visited Adams in September 1951 to discuss "programming techniques" (WW SR-27), and attendees at a significant informal discussion at a conference in December included Wilkes, Wheeler and John Bennett from Cambridge, and Adams, Carr and Jack Gilmore from MIT. Another discussant, perceiving the similarity between the two approaches, referred to a common "Wilkes-Adams philosophy" of machine organization (AIEE 1952, 113–14). Wilkes visited MIT after the conference and discussed Adams' plans for CSSR, suggesting that EDSAC's operations manager Eric Mutch be invited to MIT (Wilkes 1985, 180). Mutch duly spent the summer of 1952 at MIT contributing to the development and documentation of CSSR, while Wheeler taught on MIT's summer course (WW M-1546) and Wilkes visited briefly following an ACM meeting in Toronto (Wilkes 1985, 180). In 1953, Stanley

Gill spent the summer at MIT en route to the University of Illinois, teaching alongside Wilkes on that year's summer course (WW SR-35). He returned in the summer of 1954 and helped Adams deliver summer courses in Advanced Coding Techniques and Business Applications (WW SR-39).

Adams concluded that making programming simple enough for novices to learn with ease “essentially solves the difficult problem of where one finds a staff large enough to keep a high-speed computer busy on small-scale problems” (Adams 1952). Knowledge and experience gleaned from the EDSAC group's experience had, as Adams acknowledged, been pivotal in orienting the approach that Whirlwind's applications group took to programming after 1950. As they gained experience, however, MIT staff began to adapt and extend the EDSAC approach in various ways, as the following sections illustrate.

2. Interpretive Subroutines

By the end of 1952, then, Whirlwind was equipped with a programming environment similar in scope to EDSAC's and similarly centered on an extensive subroutine library providing easily reusable code for a range of common tasks. Both environments included a number of so-called interpretive subroutines often characterized as providing a computer with new synthetic machinery. This section describes a common use of interpretive subroutines, namely to provide programmers with a range of different number systems, an area of functionality referred to in CSSR as “programmed arithmetic.”

One of a computer's most fundamental physical characteristics is its word length, which determines the quantum of storage directly manipulable by the machine's basic operations and the range of numbers that it can handle at the highest available speed. The word length on early computers represented a compromise between the need for numerical precision and the desire to simplify machine construction, and it was widely recognized that some applications would require higher levels of precision than that provided by the hardware. An early program to test Mersenne primes run at Manchester, for example, involved the manipulation of integers of several hundred digits (Newman 1949). Increased precision was provided by using more than one machine word to store a single number, the trade-off being that additional coding was required to implement basic arithmetic operations on these numbers, making programming more complex and programs longer and hence slower.

This was a pressing issue for Whirlwind whose intended use in real-time flight simulation mandated a particularly high speed of operation. Arguing that the “short computing sequences” involved in simulation did not require great precision, its designers had chosen a 16-bit word length with a fixed decimal point

(WW SR-5, 10–1). General scientific and engineering applications would require higher precision, however, and by the end of 1947 it had been decided to provide subroutines for operations on numbers stored in two consecutive words. Special orders were included in the code in the hope that coding with these “multiple-length” subroutines would be “no more complicated than for single-length numbers” (WW SR-3). Taking the “point of view of an individual coding a problem” (WW M-221), the aim was to allow the multiple-length operations to be used “as if they were built into the computer” (WW M-111), even at the cost of longer computing times. To a greater extent than other machines, then, general-purpose use of Whirlwind was predicated on the existence of coding techniques to circumvent hardware limitations. This work became Adams’ responsibility, and by March 1948 he had drafted a set of subroutines to enable the execution of “any desired operation ... on double-length numbers” (WW SR-6).

In the summer of 1948, John Carr worked largely on what he called a “two-register” coding method (WW SR-10). As Whirlwind was a fixed-point machine, all stored numbers had an absolute value less than one and to compute with numbers outside this range programmers had to explicitly code the relevant scaling factors, a time-consuming and error-prone process. So-called floating-point methods, such as Carr’s two-register method, eased this burden by storing the scale factor as part of the number itself and automating the process of scaling. On fixed-point machines, floating-point arithmetic was provided by subroutines and involved the usual trade-off between ease of programming and speed of computation. Carr noted that his method could be extended to allow simple coding of programs that carried out computations on entities such as complex numbers, vectors, and matrices.

In the following year, Adams’ group worked mostly on the coding of specific problems, as a result of which a definitive new order code was proposed for Whirlwind. The automatic subroutine orders were eliminated and Adams revised the existing subroutines for double-length and floating-point arithmetic (WW M-887). Investigations on making the use of subroutines as easy and efficient as possible were also carried out (WW E-329). As Whirlwind approached usability at the end of 1950, however, the contacts with the EDSAC group described above led to a change of technique and so-called interpretive subroutines became central to the provision of programmed arithmetic.

John Bennett, an Australian research student working in Cambridge, developed some interpretive subroutines for EDSAC around 1949 (Campbell-Kelly 1980). Bennett’s immediate goal was to reduce the space required to store instructions so that longer programs could be written:

a considerable saving can be effected in the space occupied by orders by using a special “order” code, of which each “order” specifies a sequence

of machine orders, and by packing two such “orders” into a single storage location ... An interpretive subroutine is required to interpret “orders” expressed and packed in this form. (Wilkes, Wheeler, and Gill 1951, 162–63)

Interpretive subroutines read and decoded “pseudo-orders,” which were not part of a machine’s actual order code. Execution of a pseudo-order involved the execution of multiple machine orders and possibly the invocation of other, non-interpretive, subroutines. Saving space was thus traded off against the increase in run time caused by having the pseudo-orders decoded by software rather than hardware. EDSAC programmers soon began to use interpretive subroutines in a more general way, however, to extend or even replace a machine’s order code with new instructions, a practice that Bennett later described as a “fortuitous byproduct” of the technique (Bennett 1990).

Work on “floating-point and extra-precision interpretive subroutines” for Whirlwind started in February 1951 and that June it was reported that:

Interpretive subroutines capable of interpreting a fairly general code of instructions, resembling quite strongly the ordinary Whirlwind I order code, have been written for performing several different forms of multiple-length arithmetic. (WW SR-26)

Enlargement of Whirlwind’s memory at the end of the year made this work more urgent:

The recent availability of 1024 registers of electrostatic storage has, for the first time, made feasible the undertaking of extensive calculations which require either extra precision ... or floating point or both. (WW SR-28)

Throughout the early part of 1952 work continued on a variety of schemes. From the user’s point of view, this “programmed arithmetic” appeared to alter Whirlwind’s basic characteristics. As applications group programmer John Frankovich noted about a triple-precision interpretive subroutine he had written:

Programmers using this subroutine can assume that Whirlwind is a floating binary point computer which performs arithmetic and logical operations upon numbers greater than or equal to 2^{-64} and less than 2^{+64} with 39 binary digit precision. (WW M-1517)

As well as altering machine characteristics, interpretive subroutines could compensate for missing hardware. The utility of the index registers added to the Manchester machine in 1949 had been widely noted and they were frequently simulated on other machines:

The purpose of the early systems was to provide synthetic machines which had floating-point operations and often index registers (B-tubes), since the real machines did not. (Backus 1959, 234)

The features provided by early interpretive subroutines were thought of as extensions to the physical machine and programmers used them by switching between native and interpreted code as necessary. But no matter how similar they were to the built-in orders, a group of pseudo-orders had to be prefixed by a call to the subroutine that would interpret them, a process that required a certain level of care and sophistication on the part of the programmer. Interpretive subroutines were soon developed which removed this requirement, however, replacing all the native code with interpreted code and defining complete and self-contained synthetic machines.

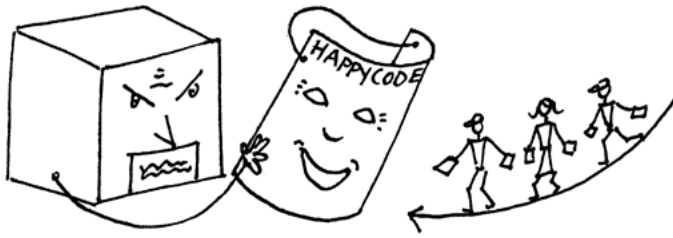
3. Synthetic Machines

MIT ran a two-week course on “Digital Computers and Their Applications” in July 1952 (WW SR-31, 17). The course was oversubscribed, attracting 95 attendees from a wide range of government and industrial organizations. Students gained practical experience of Whirlwind programming using a simplified version of the order code and special service routines that made data input and error detection easier. For the 1953 edition of the course, the applications group went one better and developed a new computer specifically designed to help students write, operate, and debug programs within the time constraints of the course. This new computer was not a physical machine, however:

The MIT Summer Session Computer, hereafter referred to as the SS Computer, does not exist as an assembly of electronic apparatus; rather its realization is achieved by appropriate conversion, assembly, and interpretive routines operating in the Whirlwind I Computer. (WW SR-35, 47)

The SS computer had its own instruction code, supported by an input routine to translate alphanumeric tapes of programs into binary form and post mortem and error diagnosis routines designed to help inexperienced programmers locate errors quickly. Development started in June 1953, and after an intense programming effort by Frankovich, Gill, and Arnold Siegel the SS computer was ready for use on the course that August. Although some errors were uncovered, it was reported that “the overall operation of the simulated computer was quite satisfactory” (WW M-2415).

The SS computer’s instruction code was translated by an interpretive routine running on Whirlwind. Unlike existing interpretive routines that extended the computer’s built-in order code with commands for specific features like floating-point arithmetic, the SS computer was realized by a single interpretive

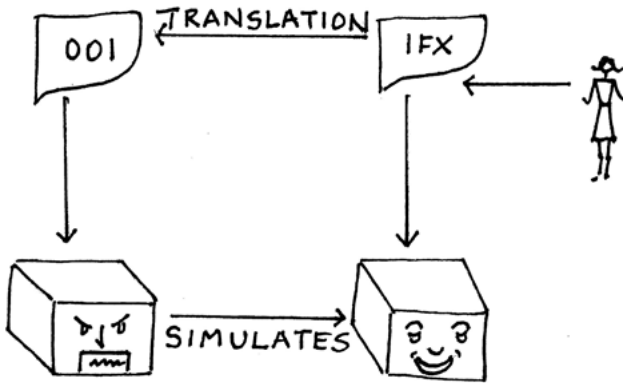


[Fig. 2] Pseudocode as a mask (after Hopper [MIT 1954])

routine which supported a complete and self-contained order code, “thus simulating a computer with characteristics quite different from those of [Whirlwind]” (WW M-2497). In place of Whirlwind’s 16-bit registers, the SS computer’s memory consisted of 28-bit words with a 4-bit tag which could store either fixed-point integers or floating-point numbers, and to enable easy modification of addresses within instructions it had seven “counters,” or index registers, a hardware feature that was entirely absent from Whirlwind. The SS computer paradoxically allowed students to use Whirlwind precisely by concealing it from them.

In the fall semester the SS computer was used on two undergraduate courses, proving to be a “very satisfactory aid” and apparently achieving its goal of making programming easier: it was reported that more than half of SS programs, all written by beginner programmers, “have given correct results on the first or second run” (WW SR-36). But this benefit came with an associated cost, as interpreted SS programs were estimated to run approximately 10 times slower than those written in Whirlwind’s native code. Low execution speed was not a critical factor in student programs but probably restricted wider use of the system, although it was used in early 1954 on a project to analyze data obtained from blast furnaces (WW M-2812, 7). The SS computer code written for this project was soon rewritten using standard CSSR routines (WW M-2870, 13), but the SS computer was used as the basis for the Single Address Computer, one of two new synthetic machines developed for a business computing course offered in the summer of 1954 (WW SR-39). The other, the Three Address Computer, provided students with access to a completely different type of machine, being “designed to simulate an actual medium speed drum computer” (WW M-2900).

By 1954 interest in automatic coding was gaining momentum, and in May the US Navy’s Office of Naval Research (ONR), which had funded much computing research (including the Whirlwind project) since the war, organized a symposium bringing together researchers in the field (ONR 1954). The ONR also sponsored a further course offered at MIT that summer, on “Advanced Programming Techniques,” organized by Adams, Gill, and Whirlwind



[Fig. 3] Interaction of the linguistic and machinic aspects of automatic programming

programmer Donn Combelle (MIT 1954). Attendees at the MIT course included Grace Hopper, who had chaired the ONR symposium, and John Backus, leader of IBM's new Fortran project.

At the MIT course, Gill described what he called the "language aspect" of automatic coding, pointing out that programmers could write programs using "pseudocodes" that were different from the instruction code of the machine and that required translation into the machine's own code before they could run. In a cartoon on the front cover of the notes from the course (Figure 2), Hopper imagined this situation from the users' point of view, depicting the pseudocode as a mask transforming a threatening and unapproachable computer into a more programmer-friendly machine.

As the example of the SS computer illustrates, such masked machines were discussed and experienced by users as if they were machines in their own right. Figure 3 expands Hopper's cartoon by making explicit the relations of translation and simulation that supported the use of synthetic machines. The diagram shows how a pseudocode could not only provide a new way to program a given computer but could also, by defining a new synthetic machine, be experienced as providing a replacement for that computer.

A few years later, Backus expanded on this idea in the context of programming languages. He defined a "synthetic machine" as one "equipped to accept statements of procedures in a language other than the 'hardware language,'" and suggested that such machines could be evaluated in the same way as "real machines," by assessing their programmability and speed of execution (Backus 1959). The consistency of the programmers' experience is guaranteed by the fact that the diagram of Figure 3 is expected to commute, in

the sense of category theory: the effect of executing a pseudocode operation on the synthetic machine is the same as the effect of running the translated operation on the simulation.

Interpretive routines were not the only way in which synthetic machines could be created. Frank Heart drew attention to the variety of practice while talking about the increasingly widespread phenomenon of synthetic computers:

Subprogramming, conversion routines, and executive routines are discussed as members of a class of methods which synthesize programmed digital computers “within” actual digital computers. Ease of programming, training, and design testing are pointed out as major reasons for turning to such procedures. (Heart 1954)

Synthetic computers were rife in the 1950s, as Hopper observed in her concluding remarks at the ONR symposium:

Miss Moser has said that “any reasonably good compiler is a pseudo-computer” ... Thus a “second-level” of computers is being “built.” Computers with a wide range of instructions, easy to program, and able to be shipped by mail are being created to exploit basic hardware. (ONR 1954, 148)

Some were “imaginary machines” with no physical counterpart, often conceived and developed like the SS computer to make programming simpler. For example, once the University of Michigan’s digital automatic computer MIDAC had been installed in 1953, instructors found that the difficulty of scaling, instruction modification, and binary representation meant that in the time available students only managed to complete trivial programs. “To alleviate these problems it was decided that a new computer was needed: one designed to make programming easier” (Perkins 1956). The resulting “pseudo-computer,” EASIAC (EASy Instruction Automatic Computer), was implemented in the same way as the SS computer: source code was translated by an input routine into an intermediate form that was then interpreted to execute the program, and a “mistake printout program” provided diagnostic information to assist in debugging.

Other developers of synthetic machines aimed to replicate physical machines. In 1958, the ACM published lists of 20–30 such efforts, many involving IBM machines as the company delivered a series of increasingly capable computers (ACM 1958a, 1958b). An older machine could be simulated on its more powerful successor as a way of preserving investment in software written for the older machine, or for other reasons. For example, the US Council for Economic and Industry Research announced a program for the IBM 704 that simulated the less powerful but widely used IBM 650. The synthetic machine was deemed valuable because “it can serve to alleviate a temporary

overload of 650 facilities, run programs with excessive storage requirements or act in many ways as a buffer for various emergency conditions" (ACM 1958a). But it could also make sense to simulate a new computer on one less powerful, for example to develop and test software for a promised new machine so that it could be put into service as soon as it was delivered. This had been the motivation behind one of the earliest applications of interpretive subroutines, in Manchester:

We are building a new machine, the code of which differs from the code of the machine now in existence ... when the new machine is finished, several months will have to be spent in getting the new routines right. We are trying to avoid this waste of time by developing a scheme which will enable us to test the new programmes on the old machine. (Bennett, Prinz, and Woods 1952)

4. Synthesizing New Kinds of Computer

Designers often aimed to make the new orders provided by interpretive subroutines and synthetic machines resemble those of existing codes. In a project run by the General Electric Company, for example,

every effort was made to keep the coding treatment of suboperation instructions as close as possible to that of [IBM] 702 instructions so that from the programmer's standpoint there would be no difference between the sixty pseudo operations made available by SCRIPT and the thirty-seven "built in" 702 commands. (Thompson 1956, 9)

A more radical proposal was to use forms of notation other than computer code, for example by allowing programming to take place directly with mathematical formulas. As early as 1949, Mauchly had proposed an interpretive "program which would accept algebraic equations as ordinarily written, which program would perform the indicated operations" (Remington Rand 1952). As implemented for UNIVAC (UNIVersal Automatic Computer) in 1952, this "Short Code" still required algebraic symbols to be hand-coded as numbers, but the third MIT system discussed at the 1954 ONR symposium allowed equations in recognizably mathematical notation to be presented directly to Whirlwind for solution.

J. Halcombe Laning, a mathematician working in MIT's Instrumentation Laboratory, visited Whirlwind in late 1951, returning early in 1952 with a proposal for a program to solve differential equations using the Runge-Kutta method. This was accepted by Adams's applications group and coded with the assistance of John Carr, using around five hours of Whirlwind runtime. In April, Laning was joined by Neil Zierler, also of the Instrumentation Lab, and the project expanded to encompass a range of programs, including a

pseudo-random number generator. Laning's original plans expanded in this context and at the end of the year it was reported that:

An interpretive program has been written to read in and solve sets of algebraic equations of a certain class, each problem being presented to the machine as a flexo-tape in essentially ordinary mathematical notation. This program has at present partial provision for the handling of more elaborate problems; e.g. iterations and the computing of special functions. (WW SR-32, 26)

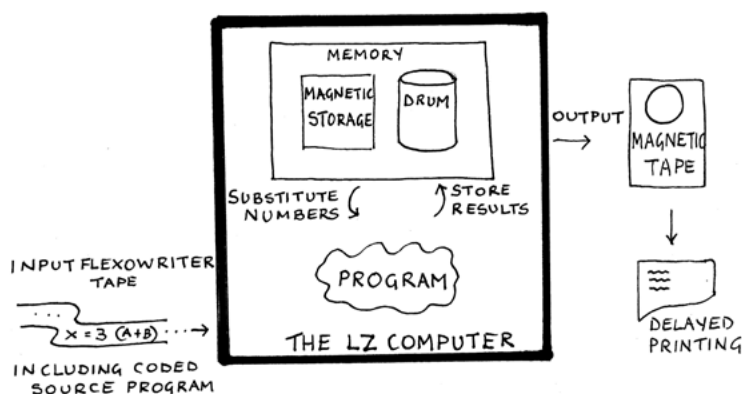
The keyboard on the Flexowriter, the teleprinter used to prepare Whirlwind's input, contained a number of mathematical symbols, including parentheses, symbols for arithmetical operations, and exponents, thus allowing complex equations such as $d = a(b + c)^2 / (c - a) a^3$ to be punched directly onto tape (Laning and Zierler 1954, 6). Equations had to be expressed as strict sequences of characters, but this format nevertheless marked a great gain in readability and usability over systems such as Short Code.

Work on the interpretive program continued throughout 1953. In November it was used on another project to compute a Fourier transform, and on December 1, Laning described it at an internal MIT seminar as an "interpretive program for mathematical equations" that

will accept algebraic equations, differential equations, etc., expressed on Flexowriter punched paper tape in ordinary mathematical notation (within certain limits imposed by the Flexowriter) and automatically provide the desired solutions. Additional features such as the automatic computation of special functions and the automatic solution of differential equations are being included. (WW SR-36, 17)

An internal report on the program was issued in January 1954 and it was used for a variety of test applications, but never gained wide usage (Laning and Zierler 1954). As with the SS computer, reports regularly stressed that programs "operated satisfactorily on their first trial" (e.g. WW M-2699), with the suggestion that the system made programming easier, but it is likely that the slow running of programs compared with other Whirlwind programming systems (a slowdown factor of 10 was often cited) hindered its wide acceptance. It was used on at least one other project during 1954, and at the end of the year Laning began an effort to rewrite it. Substantial amounts of time were dedicated to this, but by early 1955 Laning had been forced to abandon the project through pressure of other work.

The Laning and Zierler program is famous as an early, if not the first, example of a system that might be described as a programming language rather than simply an autocode. This is understandable, as its input notation looks significantly different from, say, the code of the SS computer. But there are



[Fig. 4] The structure of the synthetic machine underlying the Laning and Zierler system

also similarities between the two systems: both were implemented by interpretive routines that blocked access to Whirlwind's native code and provided the user with a synthetic machine significantly different from Whirlwind.

Laning and Zierler's report reflected both sides of the machine-language duality. While the title described the system in linguistic terms as "a program for translation of mathematical equations," the text stated that "the effect of our program is to create a computer within a computer." Laning and Zierler did not explicitly describe the synthetic machine underlying their system but its high-level structure can be inferred from their text (see Figure 4). The LZ computer, as we might call it, differed substantially from synthetic machines such as the SS computer, which shared with their host machines a familiar computer architecture organized around a single, addressable memory and were programmed with instructions consisting of operation codes and addresses.

The memory of the LZ computer held only floating-point numbers. These were not stored in addressable memory locations but, following normal mathematical practice, were understood to be the values of variables such as the a , b , c and d appearing in the example equation above. The memory was divided into two areas: "magnetic storage" and "the drum." By default, variables were stored in magnetic storage, but could be explicitly placed on the drum by means of an ASSIGN instruction.

The input Flexowriter tape contained a sequence of equations interspersed with control instructions; we will refer to this form of the program as the LZ code. Equations and instructions could be numbered to provide targets for jump instructions, and the sequence ended with a STOP instruction. Equations had the form $\langle \text{variable} \rangle = \langle \text{expression} \rangle$, and the expression on the right-hand

side was regarded as “a set of computational operations to be performed.” The number computed by these operations was “stored in the registers assigned to the letter that appears on the left hand side.” A PRINT statement allowed the values of variables to be recorded on magnetic tape for later printing in “equational form,” for example, $x = +.314157 + 01$.

Although referred to as an interpretive program, the input routine of Laning and Zierler’s system translated each equation into a block of code that was then stored on Whirlwind’s drum. This code used the CSSR routines, and when an equation had to be evaluated the corresponding code was read from the drum and interpreted in the usual way (Adams and Laning 1954, 64). However, neither this code nor the original equations were stored in the memory of the synthetic LZ computer, which held only numbers. The LZ computer therefore broke with one of the fundamental design principles of the modern computer, that of storing data and code in a single shared memory.

The LZ system thus differed from a purely interpretive system in that the LZ code, the equational form of a program, was *absent* when the program was being run: Whirlwind stored only the translated form of the program, while the memory of the LZ computer held, as described above, only numbers. However, the LZ code was at the same time *present* to the programmer and the LZ computer and used by both as a way of understanding the run-time behavior of the program. The programmer naturally imagined sequential execution of the code within the computer, each step involving the evaluation of an equation or the carrying out of an operation, and the system supported this interpretation in various ways. For example, the locations at which error conditions like overflow occurred were reported with reference to the LZ code rather than its translated form. The diagram in Figure 4 depicts this anomalous situation by showing the LZ code as an amorphous and unlocatable entity within the LZ computer but separate from its memory.

One consequence of the absence of the LZ code was that programmers could not change it while the program was running. And yet, the ability to modify a program at run time was widely recognized to be a key feature on which the utility of high-speed automatic computers rested. A form of substitution gave the LZ computer an analogous capability. Before evaluating an equation or executing an instruction, the synthetic machine substituted the variables appearing in the LZ code with their values, rounding the floating-point numbers retrieved from memory to integers when they were needed as index variables or to denote line numbers in the program. The equations and instructions of the LZ code were therefore not changed at run time, but before execution they were transformed by substitution into a form that contained no variables.

In the organization of its memory and the way it handled code, then, the LZ computer radically differed from contemporaneous hardware. Memory was not structured as a sequence of addressable words and numbers, and instructions were not stored in the same space. Numbers were stored in named variables, but the way in which instructions were stored remained undefined. The LZ code provided no way of directly accessing instructions, and these were therefore modified not by explicit programming but by a process of substitution taking place before execution. The “computer within a computer” that Laning and Zierler designed was therefore significantly different from Whirlwind, altering and in some ways restricting the familiar “von Neumann” design. Even though they were not explicit about its properties, the LZ system enabled Whirlwind to simulate a novel computer design.

Users of the LZ computer could imagine, design, and debug their programs purely in terms of the LZ code, even though that code was not directly involved in the execution of a program. In this sense, programmers worked directly with the synthetic machine rather than with Whirlwind. Nevertheless, some aspects of the underlying system remained visible, such as the floating-point number format inherited from CSSR’s programmed arithmetic. The distinction in the LZ computer’s memory between the magnetic store and the drum was of course a direct reflection of Whirlwind’s hardware and gave rise to some apparently arbitrary restrictions in the LZ input language, such as a limitation on the number of variables that could be stored in the magnetic store. There were also indirect effects on programs, such as run-time properties arising from the speed difference between magnetic storage and the drum.

In the historiography of computing Laning and Zierler’s system is usually presented as a linguistic innovation, but this was predicated upon a parallel innovation in machine design, as described in this section. This interplay between language and synthetic machine continued to play an important role in the development of increasingly sophisticated programming languages, as the following section illustrates.

5. Programming Languages

The emergence of programming languages in the second half of the 1950s has been seen as a move away from the machine towards a purely linguistic notion of programming, an interpretation visible in diagrams showing a linguistic hierarchy built on top of the physical computing machine (see Figure 1). The examples of the SS computer and the Laning and Zierler system suggest that the development of automatic programming was not simply a smooth progression upward through such a hierarchy, however. Both systems were motivated by a desire to move away from particular features of Whirlwind and both presented ideas about what a more suitable computer would be. They

did not represent a move away from the machine *in general*, but rather moves away from a *particular* machine or machine design in search of machines and notations better suited for specific purposes.

While the SS computer was programmed with a conventional machine code, Laning and Zierler's work emphasized the search for a more human-facing programming notation. Although it was not widely used at MIT it served as an inspiration for subsequent projects, as Hopper observed in 1956:

The presentation by Charles Adams of a description of Laning and Zierler's system of algebraic pseudocoding for the Whirlwind computer led to the development of Boeing's BACAIC for the 701, FORTRAN for the 704, AT-3 for the UNIVAC, and Purdue System for the Datatron and indicated the need for far more effort in the area of algebraic translators. (ONR 1956, 2)

IBM's Fortran project was started in early 1954, and the ONR symposium in May brought together researchers from MIT and IBM. Fortran project members John Backus and Harlan Herrick gave a talk on IBM's earlier Speedcoding system and listened to Adams briefly describe the LZ system. Shortly after the meeting Backus wrote to Laning, who arranged a demonstration of his system for the IBM group in June. Backus and other members of the Fortran group were also present at MIT's summer course in August, where the LZ system was discussed in more detail. Reflecting on this episode, Backus noted that for many years he had believed that Fortran had been directly inspired by the LZ system. Although he stepped back from this position somewhat, it is clear that in 1954 the IBM researchers were fully aware of Laning and Zierler's work, and there are many similarities between the two projects (Backus 1980, 129–30).

Although the Fortran project emphasized the definition of the language and its translation into efficient machine code, early documentation described Fortran as a system which "in effect transforms the 704 into a machine with which communication can be made in a language more concise and more familiar than the 704 language itself" (IBM 1956, 2). Like the LZ system, Fortran assumed a synthetic machine whose language "closely resembl[ed] the ordinary language of mathematics." The system translated a "source program" written in this language into a machine language program. In broad outline, the implicit Fortran machine was very similar to the LZ computer described above: numerical data was held in variables rather than explicitly addressed memory locations, programs were written as sequences of numbered statements that were not accessible for modification at runtime, and the variables appearing in statements were substituted with their current values before execution.

COMIT, a development led by Victor Yngve of MIT's Mechanical Translation (MT) group, provides a contrasting and less familiar example of a programming language developed in the mid-1950s. Like Laning and Backus, Yngve wanted to allow a group of non-specialists, in this case linguists rather than scientists and engineers, to do their own programming. The emerging requirements of computational linguistics, however, led to a very different language and machine model.

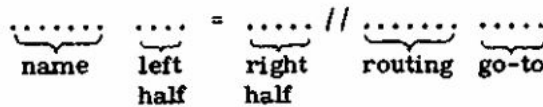
Yngve described COMIT as a solution to the problem of allowing non-experts to use the computer:

It has been the custom for the linguist who wanted to try out a certain approach to mechanical translation to ask an expert programmer to program his material rather than to learn the art of programming himself ... The linguist has not become aware of the full power of the machine, and the programmer, not being a linguist, has not been able to use his special knowledge of the machine with full effectiveness on linguistic problems.

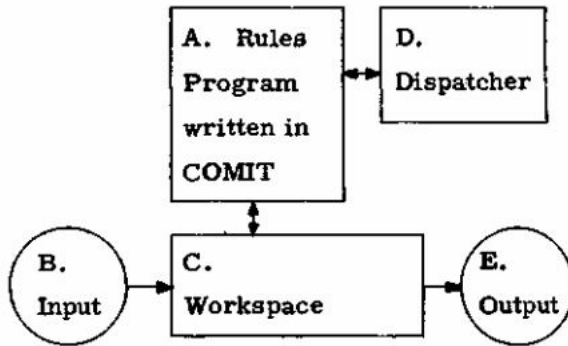
The solution offered here to these difficulties is an automatic programming system. The linguist writes the results of his research in a notation or language called COMIT, which has been specially devised to fill his needs. The programmer writes a conversion program or compiler capable of converting anything written in this notation into a program that can be run on the computer. Thus the expense, time, and effort needed to separately program each linguistic approach is saved, and, even more important, the linguist is given direct access to the machine. He becomes more fully aware of its potentialities, and his research is greatly facilitated. (Yngve 1958, 25)

To a first approximation, COMIT can be described as an attempt to take at face value Noam Chomsky's description of a grammar as "a device ... for producing the sentences of [a] language" (Chomsky 1957, 11). Yngve had criticized the prevailing "word-for-word" approach to MT, arguing that adequate translation could only be obtained by taking into account the structure of the sentence being translated (Yngve 1955). In his view translation was better modeled as a two-stage process of *decoding* the input sentence into its constituent structure and then *encoding* this structure in the target language. However, he noted that progress towards a "sentence-for-sentence" approach was hindered by the lack of an adequate account of sentence structure.

In 1955, the MIT group appointed four new staff, including Chomsky, and Yngve came to believe that Chomsky's ideas about grammar could provide a account of sentence structure adequate to ground a new approach to MT. By June 1957, Chomsky's work and the arrival of MIT's new IBM 704 computer had inspired Yngve and his colleague Hugh Matthews to begin sketching plans for a programming notation for linguists. They started working on a



[Fig. 5a] "The five sections of a rule in COMIT."



[Fig. 5b] "How a COMIT program works in the computer" (Yngve 1958)

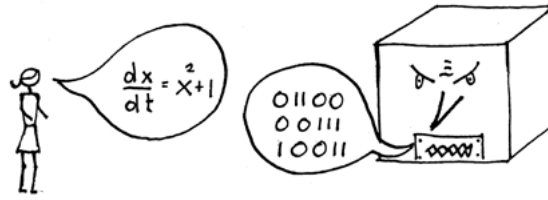
"construction routine," or program for generating sentences, but in collaboration with the programming staff of MIT's Computation Center (including Fortran veteran Sheldon Best) soon redirected their efforts towards creating a more general-purpose programming language, COMIT. Yngve published the first full description of the language in early 1958, but its implementation was not complete until 1961.

Chomsky described grammars as sets of rules capable of generating all the sentences of a language. Phrase structure rules generated what Chomsky called the "kernel sentences" of the language by rewriting strings of symbols to produce new strings:

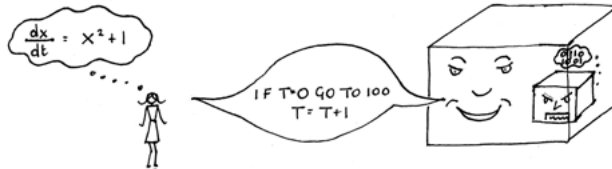
[w]e interpret each rule $X \rightarrow Y \dots$ as the instruction "rewrite X as Y."
(Chomsky 1957, 26)

Transformations operated on kernel sentences to produce more complex sentences. Transformations were also expressed as rules, but in order to determine when a transformation was applicable it was necessary to have some knowledge of the phrase structure of the kernel sentence. Transformations were therefore more complex and harder to apply than the simple production rules that described phrase structure.

Yngve and Matthews strove to organize COMIT around a single type of rule adequate to express both the phrase structure and the transformational structure of the sentences of natural language. It turned out that to make Chomsky's rules suitable for practical tasks in computational linguistics, such



[Fig. 6a] The linguistic gap between programmer and computer.



[Fig. 6b] Mediated interaction between programmer and computer

as the production of specific sentences, it was necessary to add a lot of detail and control information to the basic production rules. In their final form, COMIT rules comprised five distinct sections, as shown in Figure 5a.

As with the LZ system and Fortran, COMIT programmers could think of themselves as coding for a dedicated COMIT machine rather than for the 704 itself. Yngve described the structure of the synthetic COMIT machine in the diagram shown in Figure 5b.

The workspace was the memory holding the linguistic material being worked on. Unlike the memory of conventional computers, it was not subdivided into addressable locations. Instead, it held a single long string that was transformed by the application of rules. Like the LZ computer, the COMIT machine held its program—a set of rules—separately from the workspace in an undefined space that was not accessible to the program itself. The dispatcher held detailed information about the control flow required in the program.

With COMIT, Yngve adopted the strategy that had proved successful with algebraic and formula translation languages: notation familiar to a group of domain experts was transformed into a machine-processible form while at the same time the computer was transformed into a machine that could directly interpret the new notation and thereby be more accessible to those experts. The following section examines this double transformation from a more general perspective.

6. Programming in the Trading Zone

By the early 1960s the idea of programming languages was well established and developments were in hand for a wide range of application areas and purposes. This section considers what was distinctive about the way programming languages were thought to solve the problem of using digital computers.

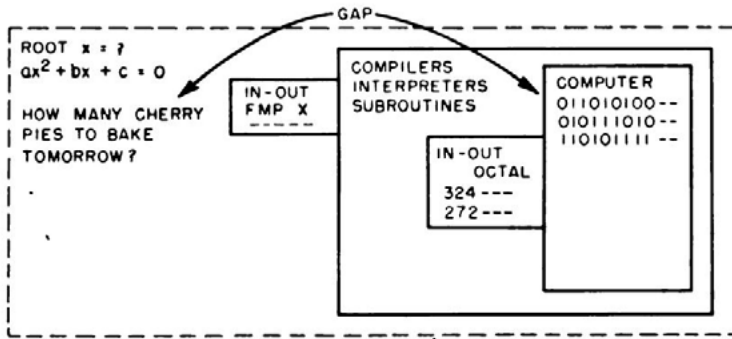
The problem was often posed in terms of a *gap* between human and machine caused by the incompatibility of human and machine language (Figure 6a), a gap that initially could only be filled by “hard, complicated, routine, clerical human labor” to translate often informally understood computations into machine code (Brown and Carr 1954, 84).

Anticipating Peter Galison’s well-known metaphor, Joseph Wegstein, a programmer at the US National Bureau of Standards, colourfully described the gap as a trading zone:

Sending a problem into a computer nowadays is like sending an expedition to Africa to trade with the natives. It has to be complete with the missionaries to translate and possibly convert the natives. If the missionaries speak only French and the native tongue then we must speak French but if the missionaries speak English too then everything is all right. (Wegstein 1956, 6)

The details of Wegstein’s analogy do not really withstand scrutiny, but as the decade progressed the gap developed into a productive transactional zone linking humans and machines. As with commercial trading zones, it was gradually bridged by the development of new technological and social *matériel* and the evolution of new means of communication. Wegstein depicted the technical innovations as a cantilevered structure of systems software reaching out across the gap from the machine towards the programmer (Figure 7).

The automatic programming systems of the early to mid-1950s mostly aimed at crossing the gap only in this direction, producing computer-based coding systems and notations that were more accessible to humans than purely binary code. But as the decade progressed, the relationship between programming languages and application areas was increasingly understood as involving more than simply making the computer easier to use. This transition is implicit in comments made by Backus in 1958 which echo Yngve’s description of COMIT as a language that linguists would find easy to use in their research:



[Fig. 7] Bridging the gap (Wegstein 1956)

Automatic programming techniques seek to make it easy to do a desired job on a computer. This is accomplished by providing a language in which the statement of the desired process is natural and concise, together with some means of causing a machine to carry out the stated process. (Backus 1959, 233)

The important point here is the idea that programming languages enable problems and processes to be expressed in notations that can be mechanically interpreted, in contrast to automatic programming systems which were designed primarily to make machines easier to use. The direction of travel across the gap has been reversed: instead of the reconfiguration of the machine, emphasis is increasingly given to the redescription of problems. RAND programmer William Orchard-Hays highlighted this point, writing that a feature of programming languages that distinguished them from other automatic coding systems was “that ‘languages’ develop around applications” (Orchard-Hays 1961, 295), and a similar thought was expressed by Bert Green, a programmer in MIT’s Lincoln Laboratory:

Digital computers have outgrown the problems for which they were designed ... Complex mathematics, business records and inventories, control of man-machine systems, and “intelligent” automatic systems are all jobs for digital computers today ... It has become necessary to develop new programming languages that are particularly suited to the more complicated problems. (Green 1961, 3)

The role of programming languages in human-machine interaction is illustrated in Figure 6b which, in contrast to Wegstein’s diagram, highlights the medial nature of programming languages as opposed to autocodes and also the double movement involved in bridging the gap: as well as transforming the computer by the provision of system software “extend[ing] the computer’s reach across the gap towards the problem,” programming languages extended

the problem towards the computer. In the context of the trading zone metaphor, programming languages can be understood as creoles: Fortran neither reflected the full complexity of mathematical notation nor allowed the fine control over computational procedure that machine code did, but it did facilitate communication and, most importantly, allow trade in the form of productive use of the computer to take place.

In general, expressing a particular application in a programming language also transformed it by causing it to be reformulated in an explicit, computable form. Computerization therefore led to epistemic innovation, as Harry Larson, editor of a special computer issue of the *Proceedings of the IRE* (Institute of Radio Engineers), noted: “when computers are applied to a field of activity outside the fields that primarily employ mathematics and logic in their theories, *that field undergoes greater formalization*” (Larson 1961, 5, emphasis in original).

As Larson pointed out, increased formalization was not particularly visible in the specific field of scientific computation, likely for the reason that decades of research in numerical analysis had already reduced large areas of mathematics to a fully computable form—and indeed automated it with both analog and digital devices. Although revisions of technique within numerical mathematics were needed to deal with the new machines’ patterns of computation, the worldview of applied mathematicians was largely unchanged by the advent of computers.

As illustrated above, however, the situation was very different in other fields such as linguistics. Although the point is somewhat controversial, it seems clear that the Chomskian revolution of the mid-1950s was at least partly informed by Chomsky’s awareness of formal logic, computability theory, and the practical development of computers (Golumbia 2009). Yngve and Matthews transformed Chomsky’s rather simple and abstract rules into a notation suitable for automation and the application of Chomsky’s ideas to tasks other than grammatical analysis.

Programming languages were not viewed solely as intermediate notations, however, but also, like the simpler autocoding systems, as systems which could materially transform the computer. Backus’s comments on programming languages quoted above continued as follows:

The synthetic machines provided by [systems like Fortran] range in speed from virtually the speed of the corresponding real machine to perhaps one-half of that speed ... Considerable effort was devoted to making the speed of the synthetic FORTRAN machine as close as possible to that of the IBM 704, its real counterpart. (Backus 1959, 234)

Discussing the origins of the Common Business-Oriented Language, COBOL, Ben Allen made a related observation, commenting that a programming

language represents “a particular *agreement* on what a digital computing machine can or should be” (Allen 2018, 25, emphasis in original). We can add that a complementary move is also involved, namely an agreement on how a particular application area should be rendered computable. Programming notations are Janus-faced, simultaneously taking up a specific perspective on the computing machine itself and also on a given application domain, a point made very early on by John Mauchly:

The particular instruction code which is selected for the machine will depend in part upon the kinds of components available for building the machine and in part upon the emphasis we place upon convenience in handling certain classes of problems. (Mauchly 1946b)

Mauchly went on to note that “there will always remain some kinds of problems for which a special machine will be better suited” than an “all-purpose machine.” In effect, the development of programming languages took Mauchly at his word, with each language postulating the existence of a synthetic machine suited for a particular application area.

7. Masking the Machine

This chapter has argued that much early work on improving programming *notation* was simultaneously understood as a way of providing programmers with better *machines*, most of which were realized through simulation rather than as distinct physical computers. These synthetic machines can be provisionally classified as follows:

1. those that alter or extend the hardware and order code of an underlying computer (see section 2);
2. those that replace an underlying computer with a simulation of a complete machine of the same type, but with different physical characteristics (see section 3);
3. those that replace an underlying computer with a simulation of a machine and order code of a significantly different type (see section 4).

This complicates the simple interpretation, given for example by Gill in 1954, of higher-level notations as purely linguistic constructs transformed into the “native language” of the computer through a process of translation. As Gill himself explained to a notable visitor to Cambridge, the computer was not singular in the machine rooms of the early 1950s:

And we were, I remember, enthusing to Claude Shannon about the horizons opened up by the idea of an interpretive routine. And I remember saying to Claude, “It’s, it’s as if we had several computers here. We have the EDSAC and then we have the floating-point computer designed by Tony Brooker which happens to run on EDSAC.” (Gill 1972, 44)

At the start of the 1960s, translation and simulation were understood to be two sides of the same coin:

Building the translating routine for [an automatic programming] language is simply the problem of constructing a routine which makes a given physical sequential machine imitate the behaviour of a nonexistent, but still sequential, machine whose language is the particular automatic coding language. (Elbourn and Ware 1962, 1060)

The interplay between these processes, depicted above in Figure 3, was described in more formal terms by Arthur Burks:

Automatic programming also involves simulation. Let U_c be a computer that operates with a machine language inconvenient for the programmer to use. The programmer uses his more convenient programmer's language. It is theoretically possible to build a machine which will understand the programmer's language directly; call this hypothetical computer M_p . Let P_t be the program (written in the language of machine U_c) which translates from the programmer's language to the machine language of U_c . Then U_c , operating under the direction of P_t , will compute the same results as M_p . That is, U_c with P_t simulates M_p . (Burks 1966, 14–15)

As Burks suggests, languages and machines occur in pairs. A programming language can be thought of as a dyad consisting of a notation and a synthetic machine on which that notation might be executed.²

It is striking, however, that as the 1950s progressed synthetic machines were less fully and explicitly described than the corresponding notations, a tendency that became ever more prominent in the 1960s. Elbourn and Ware offered an explanation for this. Designed and built as “number crunchers,” the aboriginal theory of computers was based upon decades of research into numerical analysis. But as a greater range of applications were programmed, computers came to be understood more generally as “symbol crunchers,” or information-processing machines. This called into question many existing ideas and raised the question of what the basic theory of the machines might be, if not mathematical:

Language ... is the best framework we have found for understanding what an automatic computer is functionally and what it can do ... a most important area of research for advancing the uses of computers is the study of language in the abstract. (Elbourn and Ware 1962, 1062)

- 2 The contrast between this dyadic view and the purely linguistic view perhaps fore-shadows two major approaches to programming language semantics that emerged from the 1960s onward. While denotational semantics constructed an abstract domain of mathematical meanings for programs considered purely as syntactic entities, operational approaches specified a language's meaning in terms of the behavior of an abstract machine.

The study of language in the abstract and, as compiler technology became increasingly sophisticated, an emphasis on the purely linguistic relationship of translation, appears to have drawn attention away from the complementary study of machines and simulation.

Elbourn and Ware's comments reinforce the thesis that the move towards a purely linguistic account was driven by the desire to construct a theoretical model of the computer as an object of university-based research (Nofre 2023). But the work described in this chapter had a different and more practical motivation, namely the desire to make computer programming accessible to ever wider groups, firstly to workers in mathematical sciences who were not computer experts, and subsequently to scientists in other areas and to potential computer users in commerce and industry. Far from wanting to create abstract models of the physical devices, this approach made use of novel programming notations and languages as ways to facilitate access to the physical machines to solve very practical problems.

In a general account of the emergence of computer science, Stephanie Dick has written:

From the heterogeneous assemblies of people that converged at early computer centers, several distinct computing communities emerged, each held together by a particular vision of how computers should be used ... in order to create their communities, practitioners promoted a particular image of the computer and a corresponding identity for themselves, often disassociating the new technology from certain of its inaugural associations. (Dick 2015, 60–61)

Distinct communities emerged even within the relatively narrow field of programming research in the 1950s. This chapter has described how the vision of making programming easier, of providing enough material to keep the new machines busy, generated a view of programming and the relationship between languages and machines that was distinct from the more purely linguistic, universalizing approach that emerged from concerns with portability (Nofre, Priestley, and Alberts 2014). The distinction between the two communities is subtle, as both approaches drew upon ideas about language and many actors participated in both, but it can best be appreciated by keeping in mind the differing problems being addressed. To put it another way, in the course of the 1950s, two distinct visions of programming languages emerged, one emphasizing the creation of a single, ideal, and universal language, while the other, more pragmatic approach, viewed languages as mediating specific relationships between machines and users in given application areas.

The “disassociation,” to use Dick's term, of programming from the physical machine is common to both approaches, and was noted early on:

Indeed, to many [programmers], understanding a “machine” is in reality understanding one or more programming systems for the machine, with scarcely a thought for the processor [i.e. compiler] which vivifies the system, much less for the machine itself. (Orchard-Hays 1961)

What characterized the second approach was its explicit recognition of the importance of synthetic machines and the way which it recognized that programming languages brought machines and applications together by transforming both, and in conclusion we suggest that Hopper’s suggestive imagining of a programming system as a mask captures a distinctive and flexible interpretation of programming notations which in fact rarely, if ever, allowed the underlying hardware to be completely forgotten.

At first sight, Hopper’s cartoon can be read as a straightforward depiction of abstraction, with the physical computer hidden behind the mask labeled “HAPPYCODE.” Yet this is belied by the structure of the image: the programmers are confronted not by an abstract interface, but by a machine which appears to be in the process of donning a mask. The mask is not an impermeable interface that completely obscures the machine, however, but rather something that allows it to present a different face to the world. In many situations, indeed, the function of a mask is not simply to hide something, but rather to conceal it with a view to enabling behavior and interactions that would not otherwise be possible. Surgical masks can enable the resumption of normal social behavior in a pandemic, masks worn at *bals masqués* or carnivals can license interactions that might otherwise be unacceptable, and shamanic masks are an important technique for enlarging the range of accessible spiritual or religious experiences.

It was rarely, if ever, possible to make the physical machine completely disappear, and indeed for some, like John McCarthy in the earliest stages of the development of Lisp, while the existence of an automatic programming system could shield novice programmers from the physical machine, access to it might be essential for expert use:

The source language is mainly independent of the machine being used, except that the provisions for referring directly to machine registers and their parts, which we believe must be included in any powerful source language, have been worked out only for the IBM 704 ... The conventions defining a duffers’ system might contain such a statement in order to keep the duffers uncontaminated by any actual knowledge of the machine ... Programming has not yet reached a state where all kinds of calculations can be described with no regard at all for the fact that the machine has a storage which is divided into numbered registers. (McCarthy 1957, 1, 10, 12)

In the work described in this chapter, the computer was concealed in order to make it more accessible, to allow complicated tasks to be programmed

with greater ease, and above all to enable experts from other fields to make productive use of the machine. The social dynamics powering this work were far more complex than a simple narrative of progressive abstraction would suggest, and the resulting ontology of synthetic machines complicates the familiar binary opposition between abstract and concrete.

Early versions of this work were presented at a session of the 26th ICHST conference, which took place online in 2021, and at a colloquium at the Technische Universität, Berlin, in 2022. I thank Liesbeth de Mol, Arianna Borrelli and Adrian Wüthrich for invitations to give these talks. The work on this chapter has been greatly stimulated by my participation in the ANR project “PRO-GRAMme” (ANR-17-CE38-0003-01), directed by Liesbeth de Mol. Thanks also to Karen Keogh for the hand-drawn figures.

Archival Sources

TNA: The UK National Archives, Kew, London.

WW: MIT Libraries, Project Whirlwind Collection (MC-o665).

Key to Project Whirlwind memoranda and reports cited:

- E-329: Adams, C., “Techniques for Using Standard Automatic Subroutines” (10 February 1950)
- E-516: Uchiyama, H., “Comprehensive System of Service Routines” (17 December 1952)
- M-111: Everett, R. R., “Block Diagram Work” (6 October 1947)
- M-221: Bi-weekly report (23 January 1948)
- M-887: Bi-weekly report (5 August 1949)
- M-1092: Bi-weekly report (1 September 1950)
- M-1096: Bi-weekly report (15 September 1950)
- M-1119: Bi-weekly report (27 October 1950)
- M-1517: Bi-weekly report (6 June 1952)
- M-1546: Bi-weekly report (3 July 1952)
- M-2415: Bi-weekly report (20 September 1953)
- M-2497: Bi-weekly report (2 November 1953)
- M-2699: Bi-weekly report (21 February 1954)
- M-2812: Bi-weekly report (2 May 1954)
- M-2870: Progress Report (13 June 1954)
- M-2900: Progress Report (11 July 1954)
- SR-3: Summary Report 3 (December 1947)
- SR-5: Summary Report 5 (February 1948)
- SR-6: Summary Report 6 (March 1948)
- SR-10: Summary Report 10 (July 1948)
- SR-25: Summary Report 25 (March 1951)
- SR-26: Summary Report 26 (June 1951)
- SR-27: Summary Report 27 (September 1951)
- SR-28: Summary Report 28 (December 1951)
- SR-31: Summary Report 31 (September 1952)
- SR-32: Summary Report 32 (December 1952)
- SR-35: Summary Report 35 (September 1953)
- SR-36: Summary Report 36 (December 1953)
- SR-39: Summary Report 39 (September 1954)

References

- ACM. 1958a. "Techniques Department: News." *Communications of the ACM* 1, no. 2.
- . 1958b. "Techniques Department." *Communications of the ACM* 1, no. 11.
- Adams, Charles W. 1952. "Small Problems on Large Computers." *Proceedings of the 1952 ACM National Meeting (Pittsburgh)*: 99–102.
- Adams, Charles W., and J. Holcombe Laning, Jr. 1954. "The M.I.T. Systems of Automatic Coding: Comprehensive, Summer Session, and Algebraic." In *Automatic Programming for Digital Computers*, edited by Office of Naval Research, Department of the Navy, Washington DC.
- AIEE 1952. *Review of Electronic Digital Computers: Joint AIEE-IRE Conference, Philadelphia*, December 10–12, 1951.
- Allen, Ben. 2018. "Common Language: Business Programming Languages and the Legibility of Programming." *IEEE Annals of the History of Computing* 40, no. 2: 17–31.
- Backus, John 1959. "Automatic Programming: Properties and Performance of FORTRAN Systems I and II." In *Mechanisation of Thought Processes*. National Physical Laboratory, Symposium No. 10.
- . 1980. "Programming in America in the 1950s." In *A History of Computing in the Twentieth Century*, edited by Nicholas Metropolis, Jack Howlett, and Gian-Carlo Rota, 125–35. Orlando: Academic Press.
- Barron, David. 2011. "EDSAC: A Programmer Remembers." *The Computer Journal* 54, no. 1: 139–42.
- Bennett, John M. 1990. "Early Computer Days in Britain and Australia—Some Autobiographical Snippets." *Annals of the History of Computing* 12, no. 4: 281–85.
- Bennett, John M., Dietrich G. Prinz, and Mary Lee Woods. 1952. "Interpretative Sub-routines." *Proceedings of the ACM National Meeting (Toronto)*: 81–87.
- Brown, Jeremy H., and John W. Carr. 1954. "Automatic Programming and its Development on the MIDAC." In *Automatic Programming for Digital Computers*, edited by Office of Naval Research, Department of the Navy, Washington DC.
- Burks, Arthur W. 1966. "Editor's introduction." In *Theory of Self-reproducing Automata*, edited by Arthur W. Burks, 1–28. Champaign: University of Illinois Press.
- Burks, Arthur W., Herman H. Goldstine, and John von Neumann. 1946. *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*. Princeton: The Institute for Advanced Study.
- Campbell-Kelly, Martin. 1980. "Programming the EDSAC: Early Programming Activity at the University of Cambridge." *Annals of the History of Computing* 2, no. 1: 7–36.
- Čapek, Karel. 1920. *R.U.R. (Rossum's Universal Robots)*. Page reference to the translation by Claudia Novak-Jones in P. Kussi, ed. 1990. *Toward the Radical Center: A Karel Čapek Reader*. North Haven: Catbird Press.
- Chomsky, Noam. 1957. *Syntactic Structures*. Berlin and New York: Mouton de Gruyter.
- Dick, Stephanie. 2013. "Machines Who Write." *IEEE Annals of the History of Computing* 35, no. 2: 88.
- . 2015. "Computer Science." In *A Companion to the History of American Science*, edited by Georgina M. Montgomery and Mark A. Largent, 55–68. Hoboken: Wiley-Blackwell.
- DSIR. 1949. "Department of Scientific and Industrial Research, Advisory Committee on High-Speed Calculating Machines: Minutes of the First Meeting," October 31, 1949 (TNA DSIR 10/317).
- Elbourn, Robert D., and Willis H. Ware. 1962. "The Evolution of Concepts and Language of Computing." *Proceedings of the IRE* 50, no. 5: 1059–66.
- Gill, Stanley. 1972. *Oral History Interview with Henry S. Tropp*. National Museum of American History, Computer Oral History Collection.
- Golumbia, David. 2009. *The Cultural Logic of Computation*. Cambridge, MA: Harvard University Press.
- Green, Bert F. 1961. "Using Computers to Study Human Perception." *Educational and Psychological Measurement* 21, no. 1: 227–33.

- Heart, Frank E. 1954. "The Programmed Synthesis of Digital Computers within Digital Computers." Paper presented at *ACM National Meeting (Ann Arbor)*, June 23–25, 1954. Abstract printed in *Computers and Automation* 4, no. 4: 23.
- IBM. 1956. *FORTTRAN Programmer's Reference Manual*. Applied Science Division and Programming Research Dept. International Business Machines Corporation, New York NY.
- Laning, J. Holcombe, and Neil Zierler. 1954. "A Program for Translation of Mathematical Equations for WWI." MIT Instrumentation Laboratory Engineering Memorandum E-364.
- Larson, Harry T. 1961. "The Computer Issue." *Proceedings of the Institute of Radio Engineers* 49, no. 1: 4–7.
- Mauchly, John. 1946a. "Digital and Analog Computing Machines." In *The Moore School Lectures: Theory and Techniques for Design of Electronic Digital Computers*, edited by Martin Campbell-Kelly and Michael R. Williams, 25–40. CBI Reprint Series for the History of Computing, vol. 9. Cambridge, MA: MIT Press and Tomash Publishers.
- . 1946b. "Machine Design and Instruction Codes." In *The Moore School Lectures: Theory and Techniques for Design of Electronic Digital Computers*, edited by Martin Campbell-Kelly and Michael R. Williams, 453–61. CBI Reprint Series for the History of Computing, vol. 9. Cambridge, MA: MIT Press and Tomash Publishers.
- McCarthy, John. 1957. "A Proposal for a Compiler." Internal MIT Memorandum to P. M. Morse, December 13, 1957.
- MIT. 1954. *Digital Computers: Advanced Coding Techniques*. Course Documentation, Massachusetts Institute of Technology / Summer Session.
- Newman, Maxwell H. A. 1949. "Some Routines Involving Large Integers." In *The Early British Computer Conferences*, edited by Michael R. Williams and Martin Campbell-Kelly. CBI Reprint Series for the History of Computing, vol. 14, 73–74. Cambridge, MA: MIT Press and Tomash Publishers.
- Nofre, David. 2023. "'Content is Meaningless, and Structure is All-important': Defining the Nature of Computer Science in the Age of High Modernism, c. 1950 – c. 1965." *IEEE Annals of the History of Computing* 45, no. 2: 29–42.
- Nofre, David, Mark Priestley, and Gerard Alberts. 2014. "When Technology Became Language: The Origins of the Linguistic Conception of Computing, 1950–1960." *Technology and Culture* 55, no. 1: 40–75.
- ONR. 1954. *Automatic Programming for Digital Computers*. Office of Naval Research, Department of the Navy, Washington DC.
- . 1956. *Advanced Programming Methods for Digital Computers*. Office of Naval Research, Symposium Report ACR-15.
- Orchard-Hays, William. 1961. "The Evolution of Programming Systems." *Proceedings of the Institute of Radio Engineers* 49, no. 1: 283–95.
- Perkins, Robert. 1956. "EASAC, a Pseudo-computer." *Journal of the ACM* 3, no. 2.
- Perlis, Alan J. 1983. "The Evolution of Software." Video Recording of Lecture. Computer History Museum, catalogue number 102703091.
- Remington Rand. 1952. *UNIVAC Short Code*. Remington Rand Inc., Eckert-Mauchly Division, Philadelphia, PA: October 24, 1952.
- Riskin, Jessica. 2016. *The Restless Clock*. Chicago: University of Chicago Press.
- Stibitz, George. 1943. "Capabilities of Relay Interpolator." In *A Statement Concerning the Future Availability of a New Computing Device*. Applied Mathematics Panel, NRDC, Note No. 7, November 12, 1943.
- Swade, Doron. 2011. "Inventing the User: EDSAC in Context." *The Computer Journal* 54, no. 1: 143–47.
- Thompson, Charles E. 1956. "Development of Common Language Automatic Programming Systems." In *Advanced Programming Methods for Digital Computers*, edited by Office of Naval Research, Symposium Report ACR-15.
- Turing, Alan M. n.d. [1946]. "Proposed Electronic Calculator." TNA DSIR 10/385.
- UML. 1950. "Summer School in Programme Design for Automatic Digital Computing Machines." University of Cambridge Mathematical Laboratory. TNA DSIR 10/319.

- Wegstein, Joseph H. 1956. "Automatic Coding Principles." In *Advanced Programming Methods for Digital Computers*, edited by Office of Naval Research, Symposium Report ACR-15: 3–6.
- Wilkes, Maurice V. 1950. "Automatic Computing." *Nature* 166, no. 4231: 942–43.
- . 1985. *Memoirs of a Computer Pioneer*. Cambridge, MA: MIT Press.
- Wilkes, Maurice V., David J. Wheeler, and Stanley Gill. 1951. *The Preparation of Programs for an Electronic Digital Computer*. Cambridge, MA: Addison-Wesley.
- Yngve, Victor. 1955. "Sentence-for-sentence Translation." *Mechanical Translation* 2, no. 2: 29–37.
- Yngve, Victor. 1958. "A Programming Language for Machine Translation." *Mechanical Translation* 5, no. 1: 25–41.

MACHINE TRANSLATION

COMPUTATIONAL LINGUISTICS

YEHOSHUA BAR-HILLEL

ANDREW D. BOOTH

WARREN WEAVER

Practical and Theoretical Objectives of Early Machine Translation in the 1950–60s

Jacqueline Léon

The beginning of systematic machine translation (MT) programs took place in the USA, where the first experiments started in 1949 in the context of the Cold War, and ended in 1966 with the publication of the ALPAC report. The interest in reconsidering this short initial period of about 15 years is firstly because it was rich in a profusion of ideas of all kinds, which were implemented for MT with astonishing and unprecedented enthusiasm and optimism. Secondly, it allows us to examine how the theoretical and social expectations of the actors came to be confronted with the constraints posed by the limitations of early computers and of linguistic research. Successive stances can be observed. Each of them had to cope with the issues of quality, speed, and cost. Three key personalities presided over the advent of early MT, each representing a particular angle of attack and stance: Warren Weaver, Andrew D. Booth, and

Yehoshua Bar-Hillel. In my paper, I will examine to what extent these stances were able to overcome the diverse constraints, to what extent they were opposed or, on the contrary, found some sort of agreement. Finally, one may wonder to what extent they led to the ultimate failure of MT or whether, conversely, they opened the way for new scientific fields.

This article is devoted to the early years of machine translation (MT). This period is clearly delimited by the launch of the first experiments in the USA in 1949 and by the cessation of research in 1966, in the USA and worldwide, following the publication of the ALPAC (Automatic Language Processing Advisory Committee) Report.

Since that initial period in the 1950–60s, the status of MT has evolved enormously, from experimental trial and error to become a relatively powerful tool that anyone can use on their laptop or smartphone. This evolution is due, in the first place, to the immense growth of computers and software in the 1990s, which provided large masses of data, previously impossible to store and process, but necessary for the implementation of corpus-based approaches and statistical and probabilistic methods. These methods are now largely dominant in MT, with the development of big data, deep learning, and artificial intelligence (AI). As *Le Monde* reports, the very profession of translator has been transformed. In 2018, the annual survey conducted by the European Union of Associations of Translation Companies showed that more than 50% of professional translators used an AI-based machine translation tool, such as Google Translate or DeepL (Adam 2023).

The beginnings of systematic MT programs took place in the USA, where the first experiments started in 1949 in the context of the Cold War.¹ In the scientific and military competition with the Soviets, the translation of Russian scientific texts into English was crucial. With the exception of the MT group at Birkbeck College, London, led by the British scientist Andrew D. Booth, who stayed in the USA at the end of the Second World War, research in

1 As Martin Joos (1956) indicates, the first discussions on MT took place as early as 1943 at the Signal Intelligence Service cryptography center, at Arlington Hall in Virginia, with two opposing groups: those who developed and used computers to analyze “mysterious texts” and those who translated texts from one language into another. Whereas the translators recommended their method on the basis that it was more subtle and based on intuition, and thus partially unconscious, the cryptanalysts challenged the translators to make the human process of translation understandable in order to be able to mechanize it.

other countries only began in 1954 in the wake of the first computer demonstration, which took place in January 1954 in New York on an IBM computer organized by the Georgetown University research group.² The British were the first, closely followed by the Soviets. The Cambridge Linguistic Research Unit (CLRU), headed by Margaret Masterman, started in 1954. Three Soviet groups started in 1955 (Bar-Hillel 1960; Hutchins 2013): the first in Moscow under Dmitrij Panov, the second also in Moscow under Aleksej Ljapunov, comprising Olga Kulagina and Igor Mel'čuk. The third started in Leningrad under Nikolai Andreev.³ Silvio Ceccato's Italian group also started in 1955. Other groups started in the late 1950s. In Czechoslovakia, two MT groups were created at the Charles University in Prague under Petr Sgall and Eva Hajičová. Two groups were formed by the Centre National de la Recherche Scientifique (CNRS) in France in 1959, in Paris and in Grenoble.

In addition to its chronological precedence, the USA has also been the leader in terms of MT methodology. The Soviets were aware of American work, and their own work on MT was systematically translated into English by the American JPRS (Joint Publications Research Service). Thus, with the exception of the CLRU, which from the outset developed original methods based on interlingua, thesaurus and lattice theory, the Soviet groups were initially inspired by the system developed in Georgetown before developing more original methods rooted in their national traditions (Léon 2021). Moreover, in the USSR, cybernetics and information theory, developed in the USA, were adopted as the theoretical basis of MT once the ban on them—they were considered bourgeois sciences during the Stalin era—was lifted after Stalin's death in 1953 (Mindell, Ségal, and Gerovitch 2003). It should also be noted that in the first collective book on MT, *Machine Translation of Languages*, published in 1955 and gathering 14 essays from the first conference in 1952, all the authors are American, except for the British Booth and Richard H. Richens (Locke and Booth 1955a). In other words, when we talk about the beginnings of MT, we are talking about the American work and the theoretical and ideological context in which it emerged. It was the ideas of the Americans that framed the whole field of early MT.

- 2 This demonstration of translation from Russian into English was limited to a few sentences in the field of organic chemistry, with a dictionary of 250 words and a grammar with just six rules (Dostert 1955).
- 3 Petr Petrovič Smirnov-Trojanskij (1894–1950) can also be considered a pioneer of MT, although his work in the 1930s predates the advent of computers. His project for a translation machine thought out in 1931 was only written down in 1947, in a text with the title "De la machine à traduire, construite sur la base d'une méthode monolingue." It was published in 1959 by early Soviet researchers on MT, Panov and Korolev (Panov and Korolev 1959). By taking up this 1930–40 project, the Soviet researchers' aim was to attempt to establish the Russians as the real MT pioneers before the Americans, as was the case with the launch of Sputnik in 1957.

The interest in reconsidering this short initial period of about 15 years (1949–1966) is firstly because it was rich in a profusion of ideas of all kinds, which were implemented for MT with astonishing and unprecedented enthusiasm and optimism. Secondly, it allows us to examine how the theoretical and social expectations of the actors had to be confronted with the constraints posed by the limitations of early computers and of linguistic research. MT provides an interesting case of tension between a practical goal—providing mass translations—and all sorts of constraints: theoretical views, machine capacities, quality requirements, cost limitation, and high public expectations. Many of the questions posed by the pioneers can be considered obsolete today, but others are still relevant: What is a good translation; what status should be given to a translation with errors; and what role can linguistics play in MT?

Successive stances can be observed. Each of them had to cope with the issues of quality, speed and cost, three key words which appeared in the ALPAC report summing up the various pressures on MT as a practical issue.

There is no emergency in the field of translation. The problem is not to meet some nonexistent need through nonexistent machine translation. There are, however, several crucial problems of translation. These are quality, speed, and cost. (ALPAC 1966, 16)

Three key personalities presided over the advent of early MT, each representing a particular angle of attack and stance: Warren Weaver, Andrew D. Booth, and Yehoshua Bar-Hillel. They had to work together with an almost unattainable objective at the time—given the difficulty of the task—namely to provide acceptable serial translations. In my paper I will examine to what extent these stances were able to overcome the diverse constraints, to what extent they were opposed or, on the contrary, found some sort of agreement. Finally, one may wonder to what extent they led to the ultimate failure of MT or whether, conversely, they opened the way for new scientific fields.

1. Warren Weaver (1894–1978): A Facilitator

Warren Weaver, a central actor of the culture of war, with a comprehensive view of science and a great talent for instigating projects, was the first to promote machine translation. In 1949, he sent his Memorandum *Translation* (Weaver 1955 [1949]) to approximately 200 influential people, which led to the creation of MT centers at three American universities: the Research Laboratory of Electronics (RLE) at Massachusetts Institute of Technology (MIT), the University of Washington, and the University of California, Los Angeles (UCLA). In 1958, there existed a dozen MT centers (including the RAND Corporation) in the USA. The financing of MT came from the National Air Force, the Central Intelligence Agency (CIA) and the National Science Foundation (NSF) (Bar-Hillel 1960).

Weaver's conception of MT stood at the crossroad of several ways of thinking of his time. He was a mathematician specialized in probabilities and passionate about mechanical and electric engineering. He taught mathematics at Caltech and at the University of Wisconsin, where he was trained. From 1931 until the end of his career, he was the director of the Division of Natural Sciences of the Rockefeller Foundation, where he promoted research programs in experimental and molecular biology in the USA and in Europe. During the Second World War, Weaver was part of the very powerful Scientific Research and Development at MIT, which was the very center of the military-scientific complex. He led the D-2 "fire control" section and created the Applied Mathematics Panel, gathering several hundred mathematicians to conduct studies required by the war effort and defense in extremely varied fields such as computers, programming, statistics, cybernetics, and calculating solutions to differential equations in ballistics. After the war, he returned to the Rockefeller Foundation and participated in many government institutions in charge of the organization and funding of research: the NSF, the American Association for the Advancement of Science, and the Naval Research Advisory Committee. In addition, he was one of the founding members of the RAND Corporation, developed as early as 1945–46 by the US Air Force (RAND n. d.). In his Memorandum of 1949, Weaver considered that MT corresponded to the type of objective defined by the RAND Corporation. It automated specifically human work, defined a single objective, mass produced scientific and technical translations, and implemented huge human and technical means of carrying out the objective by privileging formal solutions rather than human ones.

Weaver also played a crucial part in the promotion of information theory. In his autobiography (Weaver 1970, 110), he tells that Chester Barnard, then president of the Rockefeller Foundation, asked him to write a presentation of information theory for non-specialists in the book *The Mathematical Theory of Communication*, gathering Shannon's work and published in 1949 (Shannon and Weaver 1949). It is from this moment that the theory of information, until then confined to a few specialists in telecommunications, became known and aroused the interest of scientists of various disciplines.⁴ In *The Mathematical Theory of Communication*, Weaver mentions MT as an applied science of information theory. He suggests that there is an intercommunicability between languages that makes MT possible. This intercommunicability, Weaver says, is mainly due to invariants common to every language, among them statistical

4 Claude Shannon (1916–2001) developed his Information Theory for telecommunications at the Bell Telephone Company. The issue was to assess the most economical means of carrying the greatest number of messages through a single wire, and transmit signals with a minimum of noise. Shannon also applied information theory to cryptography: taking advantage of the redundancy property of natural languages, he used probabilistic and statistical methods to encode and decode encrypted messages.

invariants that are used for encoding-decoding in cryptography. This idea comes directly from Shannon's information theory: "a book written in Chinese is simply a book written in English which was coded into the 'Chinese code.' If we have useful methods for solving almost any cryptographic problem, may it not be that with proper interpretation we already have useful methods for translation?" (Weaver 1955 [1949], 22).

In addition, Weaver was concerned with facilitating communication between scientists as science became internationalized. It is worth mentioning that "efforts to tame language" and "to bring it under the control of science and engineer it anew for the benefit of humanity" (McElvenny 2018, 1) had dominated the cultural and intellectual field since the 1910s, notably with the revival of interest in international and auxiliary languages in the 1930s. Weaver, who was well aware of these ideas, mentions the idea of universal languages in his 1949 Memorandum. This may partly explain Weaver's somewhat lyrical and utopian formulation calling for the emergence of MT as a "new tower of Anti-Babel" in the foreword "The new tower" to the first collective book on MT (Locke and Booth 1955a):

Students of languages ..., the logicians who design computers, the electronic engineers who build and run them ... are now engaged in erecting a new Tower of Anti-Babel. This new tower is not intended to reach to Heaven. But it is hoped that it will build part of the way back to that mythical situation of simplicity and power when men could communicate freely together, and when this contributed so notably to their effectiveness. (Weaver 1955, vii)

On the linguistic level, he postulates a logical structure common to all languages (an idea he borrows from the logician Reichenbach). He hypothesizes a universal language not yet discovered, common to all languages, as well as semantic universals. But ultimately, he admits that the expertise should be left to linguists, in particular to Bar-Hillel's logical syntax (see below). In other words, Weaver's views on MT were deeply influenced by the culture of war and linked to the military-scientific complex (electronic computers, cryptography and information theory, Rand's conception of rational life), as well as by the linguistic turn and the idea of universal grammar for languages, and the interest in universal languages.

Despite all these considerations, Weaver's ideas on translation and the feasibility of MT remained rather vague and general. From the start, Weaver recognized the limits of MT, in particular the machine's incapacity to translate literary texts involving "illogical" elements like emotions and intuition. He measured linguistic difficulties such as idiomatic expressions and compound words, polysemy, and word order. That is why he confined MT to the technical and scientific texts where, he said, these problems were limited and could be

neglected. For him MT was not a discipline—it was a technology intended to produce economically profitable mass translations.

To return to the requirements of quality, cost and speed, underlying the practical utility of MT, Weaver's ambitions for the quality of translations were rather modest. He is aware that perfect translation is an illusion and that even in scientific and technical translations, which have fewer linguistic difficulties, it is impossible to guarantee the total absence of errors. However, rough and crude translations were sufficient for practical use, i.e., to allow scientists to understand articles and select those that merited human translation. In short, for Weaver, the practical use of MT must take precedence over the quality of the results.

Weaver never conducted direct research on MT and was above all a facilitator.⁵ However, he took drastic action to implement his MT project. Even before his 1949 Memorandum, which, as we have seen, led to the establishment of the first MT centers, he met several times, in 1946 and 1948, with Booth about the feasibility of MT. In 1948, on behalf of the Rockefeller Foundation, he helped him develop a magnetic digital storage system (Locke and Booth 1955b, Booth 1975). Finally, he arranged for a Rockefeller Foundation grant to be awarded to Yehoshua Bar-Hillel, who then became the first researcher in MT.

2. Booth's "Machine" Approach to MT

Andrew Donald Booth (1918–2009) was a British electric engineer, physicist, and computer scientist. He is recognized as one of the first producers of computers and a pioneer of MT in the United Kingdom. His PhD on crystallography in 1944 led him to work in the field of numerical computation (Booth 1975). Thus, in 1945 and early 1946, he started on the construction of a calculator for the automatic computation of the elements required in X-ray crystal structure analysis, at the Physics Department of Birkbeck College in London, encouraged by its director, the physicist crystallographer Desmond Bernal (1901–1971) (Birkbeck 2009). He then became interested in large-scale storage for information and identified magnetic processes as the only physical phenomena that had any potential for large-scale long-term storage (Booth 1975). From March to September 1947, Booth received funding from the Rockefeller Foundation to stay at the Institute for Advanced Study at Princeton University and worked on computers with John von Neumann and Herman H. Goldstine. During his stay, as he recalled in his 1958 book, he "produced a programme enabling an automatic digital calculator to make use of a dictionary stored in its memory-organ to facilitate the looking-up of words presented

5 He knew he was a facilitator and popularizer of science more than a real researcher (see his autobiography, Weaver 1970).

to the machine on a standard teletype input" (Booth, Brandwood, and Cleave 1958, 1).

On his return to the UK, with Kathleen Britten who was then his programming assistant—and became his wife in 1949—he began to build a magnetic digital storage system (the ARC machine, Automatic Relay Computer), based on a magnetic drum and an input-output device.⁶ In May 1948, he gave a demonstration to Weaver, who helped him with funds from the Rockefeller Foundation to develop his magnetic digital storage system to store electronic dictionaries for MT (Booth 1975).

In his Memorandum *Translation Weaver* (1949 [1955] 19) recalls that, during his visit to Birkbeck College, they discussed the feasibility of MT and that, although they both shared a keen interest in the project, they had distinct views on how to carry it out. Whereas Weaver was interested in the cryptographic aspects of translation, Booth and his colleague Richard H. Richens, Director of the Bureau of Plant Breeding and Genetics, with whom he conducted early work on MT (Booth and Richens 1955), were only interested in the mechanization and storage of bilingual dictionaries. In fact, Booth considered that the first and most feasible step for doing MT was to build dictionaries for word-for-word translation.⁷

In 1948, considering the limitations of storage—and the great expense of storage devices—Booth suggested (i) to construct dictionaries more appropriate to translation, involving not whole words but stems and endings (Booth and Richens 1955); and (ii) to divide the dictionary into two parts: first, a microglossary of words dedicated to the subject being translated, and second, a set of words of general utility. This was the project he proposed at the first MT conference, which took place at MIT in 1952.

In 1955, as he recalled in the book co-written with Leonard Brandwood and John P. Cleave (Booth, Brandwood, and Cleave 1958, 2), Booth proposed the

6 In the 1950s, as director of the Birkbeck College Computation Laboratory, Booth developed a fully electronic version of the ARC machine, SEC (Simple Electronic Computer), adopting the serial mode using shift register techniques. The shortage of electronic components was then very great, but, again with the assistance of the Rockefeller Foundation, he was able to acquire a large stock of them in order to build the first of the APE(X)C (All Purpose Electronic X-ray Computer) series of machines. Until his departure for Canada in 1962, Booth developed a large number of new machines, including MAC 1 and M2. The APE(X)C and MAC 1 machines were the first to incorporate the Booth multiplication algorithm, which gives correct results when multiplying 2's complement signed numbers.

7 Automatic translation implies a full-scale analysis of the text, so word-for-word translation very quickly proved to be completely ineffective and was replaced by the use of syntactic rules and parsers in addition to dictionaries. See below on Bar-Hillel's operational syntax.

first practical logarithmic method for dictionary searching.⁸ He did not exclude human intervention: to compensate for the limitations of what he called “the linguistic comprehension capacity of calculating machines,” he suggested that when a foreign language word was not found in the dictionary, the machine should produce it in unmodified form at the output, so that it could receive the detailed attention of the human translator. In addition, he postulated human pre- and post-editors. The pre-editor was to remove known ambiguities and difficulties from source language texts. The post-editor, on the other hand, would render the output of the machine into a respectable grammatical form of the target language. In 1958, however, Booth, together with Brandwood and Cleave (Booth, Brandwood, and Cleave 1958, 2) claimed that there was general agreement to regard these two roles as unnecessary for the MT process. In fact, as we will see, the issue of pre-, and especially post-editing would be crucial in the debates concerning the feasibility and financing of MT.

In 1958, Booth, Brandwood and Cleave provided the technical specifications necessary to build a dedicated machine for translation⁹—an idea which was rather popular among many pioneers of MT. However, as they were more interested in the internal configuration of machines, they did not seem to tackle the issue of programming tools, which, it should be said, were rather primitive at that time. It should be added that when they wrote their book in 1958, MT was just one non-numerical application among many at Birkbeck College. They were also concerned with cross-referencing of libraries, indexing of scientific and other information, telephone directories, and certain problems of mathematical logic.

Here are some technical specifications they proposed for MT:

- Some operations such as multiplication were useless for natural language processing (NLP).
- The length of the machine word must be greater than that available in contemporary machines and it would probably be satisfying to have machine instructions capable of operating on words of different lengths (Booth, Brandwood, and Cleave 1958, 296–98).
- The machine of the future should operate in serial mode rather than in parallel mode.¹⁰

8 For a dictionary of one million words, which under the second of the schemes mentioned above would take about half a million examinations to locate any given word, it was now possible by the logarithmic technique to define the position of the word in about 20 look-up operations (Booth, Brandwood, and Cleave 1958, 5). For more details on the logarithmic method for dictionary look-up, see Booth 1958, 97–98.

9 See chapter 12 of Booth, Brandwood, and Cleave (1958), dedicated to “Technical details of a proposed translating machine.”

10 “... circuits for addition in the parallel mode are somewhat complicated, and, although they are capable of great speed, a properly balanced linguistic machine would not in fact require the operations of addition and subtraction to take place at such a speed as to make these refinements necessary. This leads then to the suggestion that the machine

- Finally, concerning storage, they advocated photoscopic memory instead of the two storage organs used for calculation (the first one, a fast, generally small-sized store, used for the actual performance of calculation, and the second one, a slow backing store, often on a magnetic drum).¹¹

As linguistic data should not be erased, photoscopic memory of greater capacity should make it possible to store them in a permanent form.¹²

At Birkbeck College, in addition to electronic computers, Booth and his colleagues developed several types of computer tools for MT: dictionaries, algorithms for looking up words in dictionaries, and specifications for a translating machine (Booth, Brandwood, and Cleave 1958). As to his conception of MT itself, i.e., word for word translation with post-editing, he tolerated rough translation output with potential errors.

In conclusion, Booth's approach was mainly concerned with the practical and technological aspects, that means the computer aspects of MT, as he explicitly stated:

So much for purely philosophical views of translation, which are hardly likely to find any general measure of agreement either among linguists or among students of ideas. We proceed to a more special consideration which is bounded on the one side by what is useful and on the other by what is practicable (Locke and Booth 1955b, 1).

3. Yehoshua Bar-Hillel (1915–1975): Practical and Theoretical Reflections on the Status of Translation and Machine Translation

Bar-Hillel was a logician, a language philosopher, an inheritor of the first mathematization of language, and a fervent follower of German philosopher Rudolf Carnap.¹³ He was recruited to MIT in 1951 as the first full-time researcher in MT

of the future might work in the serial mode (Booth, Brandwood, and Cleave 1958, 296–97).

- 11 The photoscopic memory system, developed first by Gilbert King in 1953, was a device for rapid access to large quantities of stored information. It consisted of a glass disc which "acts as an automatic dictionary which is read by means of a cathode-ray tube light source, a moving lens, a photo multiplier tube, and some electronic circuitry (the digit detector)" (IBM 1959).
- 12 "The magnetic drums of the present machines may be replaced by some form of photographic storage in the machine of the future. Photographic storage has considerable attractions for linguistic use, most of which stem from the facts that it can be inspected visually, that it is permanent, and that its volumetric efficiency is even higher than that attainable on a magnetic drum" (Booth, Brandwood, and Cleave 1958, 297).
- 13 The first mathematization of language in the 1930s can be defined by the rise of formalization promoted by the Vienna School of Logics, in particular Carnap, as a common

on a Rockefeller grant thanks to Weaver. At all the steps of his commitment in MT, along with theoretical interests he showed practical concerns regarding the provision of mass translation by taking into consideration its feasibility as a computer application. This can be seen in his first publication on the topic, *The Present State of Machine Translation* (Bar-Hillel 1964 [1951]), summarizing his positions at that early stage:

Practical interest in MT arose with the desire for very rapid, low accuracy, provisional translation of large amounts of general material in foreign languages, as well as for high accuracy translation, mainly of scientific material. On the other hand, engineers have been impressed by the possibilities inherent in various components of electric computers and are now looking for further practical applications in noncomputational fields of new combinations of these components. MT seems to them to be promising since its basic logical operations are of a kind already carried out in existing computers. (Bar-Hillel 1964 [1951], 164)

This can also be seen in his evaluation report *The Present Status of Automatic Translation of Languages*, commissioned by the NSF and published in 1960, which starts with considerations on the financial and human costs of MT:

Machine translation has become a multimillion dollar affair. It has been estimated that in the United States alone something like one and one-half million dollars were spent in 1958 upon research more or less closely connected with MT, with approximately one hundred and fifty people, among them eighty with MA, MSc or higher degrees, working in the field, full or part time. (Bar-Hillel 1960, 92)

At the end of the MIT conference of 1952, there was general agreement that it should be possible to proceed toward translation by machine in two steps.

Step 1. An operating automatic dictionary can be developed within 12 months provided grammatical problems, such as case ending, verb forms and idiomatic phrases, were left aside. Magnetic storage with suitable input, searching, and output equipment can be simultaneously constructed.

Step 2. Operational analysis of syntax with a view to programming in terms of machine operations can be started immediately. Grammatical and syntactical problems can be postponed until later. Operational syntax was proposed by Bar-Hillel resulting from his theoretical views on logical syntax inherited from Carnap, which he would publish in the journal *Language* in 1953 in an article titled “A Quasi-Arithmetic Notation for Syntactic Description” (Bar-Hillel 1953). Operational syntax made it possible for a machine to automatically identify

objective for every science, setting up mathematics as one language among others. The first mathematization of language is characterized by the setting in interaction of algorithms and formal languages resulting from mathematical logic.

the syntactic structure of strings in natural language texts.¹⁴ It also paved the way for the development of algorithms that could translate the syntactic description of the language into a series of digital computer instructions.¹⁵

Through this agreement on two steps, we can see a compromise between Booth's views on dictionaries and storage, and Bar-Hillel's syntactical approach: for him, word-for-word translation was totally ineffective and completely wrong from the point of view of its theoretical conception. The expected output was machine-aided translation, which supposed the human intervention of a post-editor to make the translation acceptable. This view was widely shared by MT actors, as mentioned above. However, Bar-Hillel was the only one to justify machine-aided translation theoretically, and not only for practical reasons. He warned against what he called "fully automatic high quality translation" (FAHQT). His arguments, developed from 1951 and up to his evaluation report published in 1960, include the definition of what a good translation is. For him, at that time, a high quality translation is "a translation of the quality produced by an experienced human translator."¹⁶ In 1951, Bar-Hillel considered that FAHQT was unattainable because the machine was unable to solve semantic ambiguities.¹⁷ Later, especially in Appendix III of his report of 1960, he developed theoretical arguments. See his demonstration:

There exist extremely simple sentences in English ... which, with certain linguistic contexts, would be uniquely (up to plain synonymy) and unambiguously translated into any other language by anyone with a sufficient knowledge of the two languages involved, though I know of no program that would enable a machine to come up with this unique rendering

- 14 In this article, Bar-Hillel combined different American distributionalist methods, most notably that of Zellig Harris, as presented in his book *Methods in Structural Linguistics* (1951), and Ajdukiewicz's notation (1935).
- 15 Programmers of these algorithms sought to automatically test the syntactic connectivity of a given sequence and find the immediate constituents of any syntactically related sequence. Besides, automatic operational syntax in the phase of post-editing was considered to help human post-editors eliminate the syntactic ambiguities (Bar-Hillel 1960, 146).
- 16 Later, Bar-Hillel's view changed. At the MT conference of 1971, mainly dedicated to the quality and validity of translations, Bar-Hillel states: "Every program for MT should be immediately tested as to its effects on the human user. He is first and final judge and it is he who will have to tell whether he is ready to trade quality for speed, and to what degree." The question of the quality of translation related to user need would become dominant in the next decades (Wilks 1992, 180).
- 17 Others, like Victor Yngve, using the term "semantic barrier," pointed at the incapacity of machines to understand: "Work in mechanical translation has come up against a semantic barrier... we have come face to face with the realization that we will only have adequate mechanical translation when the machine can 'understand' what it is translating and this will be a very difficult task indeed ... 'understand' is just what I mean ... some of us are pressing forward undaunted" (Yngve 1964, 279).

(unless by a completely arbitrary and ad hoc procedure whose futility would show itself in the next example). (Bar-Hillel 1960, 158)

He gave the following example of such ambiguous sentences, where “pen” has the double meaning of “pencil” and “playpen”:

The box was in the pen in the linguistic context

Little John was looking for his toy box. Finally he found it. The box was in the pen. John was very happy.

To solve these ambiguities and choose the right meaning of “playpen,” the computer needs extralinguistic knowledge, i.e., that a pencil cannot contain a box. Yet, contrary to humans, computers don’t have the capability of understanding common sense knowledge. However, again, he manifested insights for the future: in the case of ambiguities unsolved by syntactic or semantic procedures, when computers would be able to treat masses of data, statistics would be useful to choose the more frequent use in the target language:

[I] entertained hopes that by exploiting the redundancy of natural language texts better than human readers usually do, we should perhaps be in a position to enable the computers to overcome, at least partly, their lack of knowledge and understanding. (Bar-Hillel 1964, 212–13)

Besides, while in 1960 Bar-Hillel regarded learning machines as potential devices for eventually resolving these kinds of ambiguities,¹⁸ four years later, within the context of the early research on AI, he came to the conclusion that “even machines with learning abilities, as we know them today or foresee them according to known principles, will not be able to improve by much the quality of the translation output” (Bar-Hillel 1964, 212).

In addition, the quality of the machine output is absolutely required, contrary to Weaver, Booth and most MT groups who claimed that a percentage of errors was acceptable when practical use was at stake.¹⁹ This argument raises the question of a technique which would be irremediably fallible by definition

18 In fact, I don’t know who was the first to have insight into learning machines. One presentation on learning machines was scheduled at the first conference on MT in 1952, but the expected speaker, Dr. Louis N. Ridenour, was absent. See also Yngve, who mentioned learning machines with feedback, probably referring to cybernetics: “We might say that the man is providing feedback of the type required for learning and that he is altering the machine in such a way that it behaves as if it were learning by its mistakes” (Yngve 1954, 22).

19 Even recently, this argument has constantly been put forward by the actors of MT and NLP, although the status of such errors is ambiguous. Drawing on Loffler-Laurian’s arguments (1996), we can ask: What does an 80% satisfactory translation actually mean? Does it mean that 80% of the sentences look like sentences written by humans, or that 80% of the words were given a correct equivalent, or that the translation was considered acceptable by 80% of people consulted? It is a fundamental question for MT and for NLP in general.

and that could not be improved. When radar, and other technologies developed within the framework of the sciences of war, did not work, one tried to improve them.²⁰ Bar-Hillel was totally opposed to the compromise consisting of providing an output with 5% errors. For him, in addition to the quality of translations flawed with errors, it is the trustworthiness of MT that must be addressed. Bar-Hillel claimed that there are cases of wrong output of MT that are completely grammatical and make sense, so errors are hidden and remained undetected by the user. In other words, by accepting errors, it is the reliability of the machine which is at stake.

With such requirements of output quality, machine reliability, and yet the impossibility of FAHQT, only MT with human post-editing was feasible.

It should be said that, in the following quotation, also extracted from his 1960 report, Bar-Hillel admits that situations exist where lower quality machine output may be satisfactory. He here refers to scientific texts, among others. This indicates that Bar-Hillel keeps in mind that MT is aimed at practical uses, above all:

Those who are interested in MT as a primarily practical device must realize that full automation of the translation process is incompatible with high quality. There are two possible directions in which a compromise could be struck: one could sacrifice quality or one could reduce the self-sufficiency of the machine output. There are many situations where less than high quality machine output is satisfactory ... if however high quality is mandatory ... then the machine output will have to be post-edited, thereby turning, strictly speaking, MT into *machine aids to translation*. (Bar-Hillel 1960, 95, italics in the original)

Besides, MT with post-editing was the only solution that would be commercially competitive with existing human translation:

The only reasonable aim, then, for short-range research into MT seems to be that of finding some machine–post-editor partnership that would be commercially competitive with existing human translation, and then to try to improve the commercial effectiveness of this partnership by improving the programming in order to delegate to the machine more and more operations in the total translation process which it can perform more effectively than the human post-editor. (Bar-Hillel 1960, 99)

At this stage, Bar-Hillel, like many others, had left aside pre-editing, which in fact could be avoided by translating texts in restricted domains with limited terminology.²¹

20 For opponents of the ALPAC report, this argument was used to claim that MT had not had enough time to improve and to prove it could work (Josselson 1971).

21 See also Yngve (1954).

One of the reasons why we do not as yet have any translation centers ... is the reluctance of many MT workers to recognize that the idea of inventing a method for fully automatic high quality translation (FAHQT) is just a dream which will not come true in the foreseeable future. By not realizing the practical futility of this aim, ... they have misled themselves and the agencies which sponsored their research into not being satisfied with a partly automated translation system whose principles are well understood today, and instead to wait for the real thing which was believed, and made to believe, to be just around the corner. (Bar-Hillel 1960, 158)

In particular, Bar-Hillel was very harsh on all of Erwin Reifler's claims.²² Reifler, of Seattle University of Washington, alleged to have developed the most advanced translation system, using "The USAF Automatic Language Translator Mark I" (Reifler 1954). Bar-Hillel criticized the very name of Mark I, that seemed to indicate that it was a specific device for MT, which was false.²³

In order to fight this position, which nourished the illusory public and media expectations of a forthcoming FAHQT, Bar-Hillel not only promoted machine-aided translation but also attempted to separate theoretical linguistics from MT—probably in order to save it from a predicted disaster.

Let's conclude this section by saying that Bar-Hillel put quality before other criteria like speed and cost, which implied the use of syntactic parsing in addition to bilingual electronic dictionaries. It is because he was fully aware of the weak capabilities of computers that he advocated human post-editing. Let's add that his concern about machine understanding was a real theoretical issue.

22 "For reasons given above, I must strongly disagree with Reifler's 'belief that it will not be very long before the remaining linguistic problems in machine translation will be solved for a number of important languages.' How dangerous such prophecies are is illustrated by another prophecy of Reifler's, to the effect that 'in about two years (from August 1957) we shall have a device which will at one glance read a whole page and feed what it has read into a tape recorder and thus remove all human cooperation on the input side of the translation machines.' The best estimates I am aware of at present mention five years as the time after which we are likely to have a reliable and versatile print reader (section 1.3) at the present rate of research and development" (Bar-Hillel 1960, 103).

23 The USAF Automatic Language Translator Mark I was built in 1958 by Gilbert King for the United States Air Force to translate Russian into English. In fact, it was a mechanical bilingual dictionary dedicated to translation. It was based on a high-capacity, fast-access photoscopic memory, storing the correspondences (in effect, the inflected forms) between the source and target languages and allowing rapid searching throughout the memory (or dictionary) (Shiner 1958). Thus, it was a dictionary and not a translation machine capable of carrying out the entire translation process, hence Bar-Hillel's criticism.

4. The ALPAC Report and the End of the First MT Period

In the wake of Bar-Hillel's recommendations, the ALPAC Committee was set up by the National Academy of Sciences and the National Research Council in 1964 to evaluate the performances of MT. Its conclusions, published in the ALPAC report in 1966, led to the termination of all the experiments in the USA, and put a stop to many MT projects in the world. However, it can be claimed that ALPAC was not that "infamous report" as qualified by actors of MT (Hutchins 1996). In fact, its conclusions made it possible to settle the domains of computational linguistics and NLP on more solid foundations (see Léon 2021).

It was probably the first time that such a systematic evaluation as the one developed by the ALPAC was set up in the scientific field, most likely in the wake of evaluations carried out in the entire military-industrial sector at the time. Instead of founding their evaluation on the reports provided by the MT groups as Bar-Hillel did, the ALPAC report was based on a systematic evaluation. The experts developed their own methods based on experiments and tests (statistics, psychotechnics etc.). In addition, taking into consideration the criteria of quality, speed and cost, they compared post-edited machine translation with human translation and the cost of training researchers in foreign languages.

The ALPAC report concluded that there was no shortage of translators in the USA, contrary to what was claimed by MT supporters. Besides, machine translation with post-editing was slower and more costly than human translation. Finally, the quality of machine translation without post-editing was very poor. Hence, the ALPAC stated that MT was not useful²⁴ and that experiments should not be funded any more. It was then unproductive to spend considerable amounts of money to mechanize such a small industry:

...it is ironic that several agencies propose to spend more money for "machine translation." The Committee is puzzled by a rationale for spending substantial sums of money on the mechanization of a small and already economically depressed industry with a full-time and part-time labor force of less than 5,000. (ALPAC 1966, 12)

However, even if it is absolutely true that its conclusions led to the cessation of MT experiments in the United States, the ALPAC report also had positive

24 Victor Yngve of the MIT Research Laboratory of Electronics, in answer to a request from Committee Chairman John R. Pierce, expressed his views as follows: "I concur with your view of machine translation, that at present it serves no useful purpose without postediting, and that with postediting the over-all process is slow and probably uneconomical" (ALPAC 1966, 24).

effects. And it should be said that many of these positive effects originated from Bar-Hillel's ideas.

Following Bar-Hillel mentioning this point in his report, the committee examined the impact of MT on other sciences and technologies.²⁵ They saw low impact on computer hardware. In particular they established that photoscopic memory and machines especially dedicated for translation (which was Booth's idea among others) were unsuccessful. By contrast, computer software benefited from MT, with the emergence of programming languages such as COMIT (1962), SNOBOL (1964), and ALGOL (1958, 1960) bringing into play formal grammars, in particular Chomsky's type 2 grammar²⁶ (see Appendix 18 of ALPAC report):

The period from ALGOL to the present shows intense borrowing of current mathematical linguistic theory, technique, and notation. The source of this dependency can be traced to the definition of ALGOL 60 syntax production notation. The similarity between this notation and the rewriting rules of some linguistic models caused this theory to be rapidly employed in programming. Still, it is important to note that the definition of the ALGOL language was totally inspired by programming considerations (Fortran, LISP), and not linguistic ones.

The effect of this syntax formalism has been enormous and all to the good. Thus ALGOL syntax is "essentially" of Type 2. Hence, parsing mechanisms for Type 2 languages can be applied in the construction of ALGOL translators. Many of the parsing techniques employed were, however, discovered by programmers operating in parallel to, but independent of, similar developments in mathematical linguistics. (ALPAC 1966 Appendix 18, 118)

The dependence of COMIT on formal grammars is close and direct. It was described by its author, Victor Yngve, head of the MT group and director of the MIT Research Laboratory of Electronics, as a language "specially designed to be convenient for the linguist so that he can do his own programming" (Yngve 1958, 25).²⁷

25 In its concluding remarks, the ALPAC report (1966, 32) states: "We have already noted that, while we have machine-aided translation of general scientific text, we do not have useful machine translation We have noted that the important contributions of MT have been primarily to linguistics and secondarily to computer programming. We have noted that while translation itself is vital, needs for translation are being met by a small though capable activity."

26 See Chomsky's hierarchy of grammars (Chomsky 1956).

27 It is unfortunate and surprising that in the collective book edited by Leslie Fox (1966) and devoted to non-numerical computing, COMIT is not mentioned at all. Moreover, MT and more generally NLP occupy only half a page in chapter 6, devoted to "a survey of non-numerical applications," and only three pages in chapter 9, devoted to "information

Finally, linguistics benefited from MT with the emergence of computational linguistics. In its recommendations, the ALPAC separated discredited MT from the rest of NLP (which nevertheless included machine-aided and human-aided translation). For this purpose, they created a new term “computational linguistics,” which included NLP and theoretical linguistics. By theoretical linguistics, the ALPAC meant and promoted the Chomskyan program. This idea, remember, was suggested before by Bar-Hillel in his report, to make a distinction between theoretical linguistics and MT. In addition, the members of the committee seemed to agree with Bar-Hillel on the fact that linguistics should be evaluated and financed as an autonomous discipline.

This aspect is developed in Appendix 19. In this section of ALPAC (1966, 121), it is recalled that Igor Mel’čuk, a Russian MT pioneer from the Ljapunov group in Moscow (see above), considers that computational linguistics is not a field of linguistics, a subspeciality for those who like computation; it is a technique inescapable for any linguist who honors his discipline. This is a view shared by several Russian researchers.²⁸ In Akhmanova et al. (1963, 46), we read: “MT is simultaneously both a workshop, where the methods of precise linguistic research are perfected independently of the concrete sphere of application of these methods, and an experimental field, where the results are verified by experience.”

5. Conclusions

In conclusion, it can be said that the MT experiments of the pioneer period, although discredited by various evaluations, the Bar-Hillel report and then the ALPAC report, which led to the disappearance of MT centers as such in the USA, were not without results in the fields of computer science, linguistics, and NLP. Indeed, despite, or rather because of, their triviality, the criteria of quality, cost and speed used to evaluate MT as a practical industry of mass translation have been responsible for the creation of a new field, computational linguistics, and have contributed greatly to the autonomy of linguistics.

Another interest lies in the fact that a practical objective such as MT allowed three researchers—Warren Weaver, Andrew D. Booth, and Yehoshua Bar-Hillel—with such different backgrounds and conceptions, to work together and to forge a common culture for formal linguistics and automatic language processing, opening the way to the second mathematization of language.

retrieval and some related computer problems.” It is true, however, that in 1966, after the publication of the ALPAC report, MT was no longer in vogue.

28 It should be remembered (see above) that, just as American MT work was well known to the Russians, work done in the USSR was systematically translated into the JPRS collection and known to American researchers.

It can be assumed that the second mathematization of language began with the implementation of parsing methods for MT. In other words, formal languages, rooted in mathematical logic in the 1930s–40s, characterizing the first mathematization of language, became involved in algorithms and programming for syntactic analysis. This development determined the rise of formal grammars, in particular Chomsky's generative grammar.

I would like to thank the participants in this book for their valuable comments and criticisms that helped to improve this article, in particular Troy Astarte, Arianna Borrelli, and Helena Durnová.

References

- Adam, Louis. 2023. "Illustrateurs, codeurs, traducteurs: ces métiers déjà bousculés par l'IA." *Le Monde*, January 22–23.
- Ajdukiewicz, Kazimierz. 1935. "Die Syntaktische Konnexität." *Studia Philosophica* 2: 1–27.
- Akhmanova, Olga S., Igor A. Mel'čuk, Revekka M. Frumkina, and Elena V. Paducheva. 1963. *Exact Methods in Linguistic Research*. Berkeley: University of California Press.
- ALPAC. 1966. *Language and Machines: Computers in Translation and Linguistics. A Report by the Automatic Language Processing Advisory Committee (ALPAC)*. National Academy of Sciences: National Research Council.
- Bar-Hillel, Yehoshua. 1953. "A Quasi-Arithmetic Notation for Syntactic Description." *Language* 29: 47–58.
- . 1960. "The Present Status of Automatic Translation of Languages." *Advances in Computers* vol. 1, edited by Franz C. Alt, 91–141. New York and London: Academic Press.
- . 1964 [1951]. "The State of Machine Translation in 1951." In *Language and Information*, 153–65. Reading, MA: Addison-Wesley Publishing Company.
- . 1964. "Four Lectures on Algebraic Linguistics and Machine Translation. Fourth Lecture: Why Machines Won't Learn to Translate Well." In *Language and Information*, 185–222. Reading, MA: Addison-Wesley Publishing Company.
- Birkbeck. 2009. "Obituary: Professor Andrew Booth." Accessed October 11, 2024. <https://www.bbk.ac.uk/about-us/obituaries/obituary-professor-andrew-booth>.
- Booth, Donald Andrew. 1975. "Computers in the University of London 1945–62." *The Radio Electronic Engineer* 45-47: 341–45.
- and Richard H. Richens. 1955. "Some Methods of Mechanized Translation." In *Machine Translation of Languages, 14 essays*, edited by William N. Locke and Andrew D. Booth, 24–46. Cambridge, MA and New York: MIT Press and John Wiley.
- , Leonard Brandwood, and John P. Cleave. 1958. *Mechanical Resolution of Linguistic Problems*. New York: Academic Press.
- Chomsky, Noam. 1956. "Three Models for the Description of Language." *IRE (Institute of Radio Engineers) Transactions on Information Theory* IT-3: 113–24.
- Dostert, Léon. 1955. "The Georgetown-I.B.M experiment." In *Machine Translation of Languages, 14 essays*, edited by William N. Locke and Andrew D. Booth, 124–35. Cambridge, MA and New York: MIT Press and John Wiley.
- Fox, Leslie, ed. 1966. *Advances in Programming & Non-numerical Computation*. Oxford: Pergamon Press.
- Harris, Zellig Sabbetai. 1951. *Methods in Structural Linguistics*. Chicago: The University of Chicago Press.
- Hutchins, William J. 1996. "ALPAC: The (In)famous Report." *MT News International* 4: 9–12.
- . 2013. "Historical Sketch of Machine Translation in Eastern and Central Europe." ACL Anthology. Accessed October 11, 2024. <https://aclanthology.org/www.mt-archive.info/10/Hamburg-2013-Hutchins.pdf>.

- IBM. 1959. *Final Report on Computer Set AN/GSQ-16 (XW-1). Volume 1: The Photoscopic Memory System*. RADC-TR59-110 June 20, 1959. IBM Research Center, Yorktown Heights, NY.
- Joos, Martin. 1956. "Review of William N. Locke and Andrew D. Booth (eds). *Machine Translation of Languages: Fourteen Essays* (1955)." *Language* 32, no. 2: 293–98.
- Josselson, Henry H. 1971. "Automatic Translation of Languages Since 1960: a Linguist's View." *Advances in Computers* 11: 1–58.
- Léon, Jacqueline. 2021. *Automating Linguistics*. Cham: Springer.
- Locke, William N., and Andrew D. Booth, eds. 1955a. *Machine Translation of Languages, 14 Essays*. Cambridge, MA and New York: MIT Press and John Wiley.
- . 1955b. "Historical Introduction." In *Machine Translation of Languages, 14 Essays*, edited by William N. Locke and Andrew D. Booth, 1–14. Cambridge, MA and New York: MIT Press and John Wiley.
- Loffler-Laurian, Anne-Marie. 1996. *La traduction automatique*. Lille: Septentrion.
- McElvenny, James. 2018. *Language and Meaning in the Age of Modernism: C.K. Ogden and His Contemporaries*. Edinburgh: Edinburgh University Press.
- Mindell, David D., Jérôme Ségal, and Slava Gerovitch. 2003. "Cybernetics and Information Theory in the United States, France and the Soviet Union." In *Science and Ideology: A Comparative History*, edited by Mark Walker, 66–95. London: Routledge.
- Panov, Dimitri I., and Lev N. Korolev, eds. 1959. *La machine à traduire de P.P. Trojanskij*. Édition de l'Académie des sciences de Moscou. French translation CASDN, n° T/R/ -1059.
- RAND. n. d. *A Brief History of RAND*. Accessed October 11, 2024. <https://www.rand.org/about/history.html>.
- Reifler, Erwin. 1954. "Report on the First Conference on Mechanical Translation." *Mechanical Translation* 1, no. 2: 23–32.
- Shannon, Claude E., and Warren Weaver. 1949. *The Mathematical Theory of Communication*. Urbana-Champaign: University of Illinois Press.
- Shiner, George. 1958. The USAF Automatic Language Translator Mark I. *IRE National Convention Record*, Part 4, 296–301.
- Weaver, Warren. 1955 [1949]. "Translation." In *Machine Translation of Languages, 14 Essays*, edited by William N. Locke and Andrew D. Booth, 15–23. Cambridge, MA and New York: MIT Press and John Wiley.
- . 1955. "The New Tower." In *Machine Translation of Languages, 14 Essays*, edited by William N. Locke and Andrew D. Booth, v–viii. Cambridge, MA and New York: MIT Press and John Wiley.
- . 1970. *Scene of Change: A Lifetime in American Science*. New York: Charles Scribner's Sons.
- Wilks, Yorick. 1992. "Systran: It Obviously Works But How Can It Be Improved?" In *Computers in Translation, a Practical Appraisal*, edited by John Newton, 166–88. London: Routledge.
- Yngve, Victor H. 1954. "The Machine and the Man." *Mechanical Translation* 1, no. 2: 20–22.
- . 1958. "A Programming Language for Mechanical Translation." *Mechanical Translation* 5, no. 1: 25–42.
- . 1964. "Implications of Mechanical Translation Research." *Proceedings of the American Philosophical Society* 108, no. 5: 275–81.

SCIENTIFIC COMPUTING

PARTICLE PHYSICS

CERN

MAURICE WILKES

COMPUTING IN THE 1950S

Big Machines for Big Science: The Beginnings of Scientific Computing at CERN

Arianna Borrelli

This paper offers a broadly contextualized analysis of the process leading to the purchase of the first electronic digital computer at the European Organization for Nuclear Research (CERN) in Geneva in 1956. I will discuss in detail relevant archival material and sketch the context of the events in its scientific, technological, commercial, institutional, national, and cultural aspects. This will show how CERN's decision to acquire a computer was motivated neither by the need for computational power for specific scientific goals nor by good marketing on the part of computer manufacturers. The purchase of a computer, and specifically of a Ferranti Mercury, resulted from a combination of various situated factors, which ranged from personal and institutional contacts to the vague but powerful vision of how computers could be turned into scientific instruments for particle physics. The reconstruction provides insights into the emergence

of a discipline-specific culture of scientific computing in high-energy physics comprising not only machines and codes, but also programmers, users and networks for shaping and sharing resources and know-how according to common methods and aims.

1. Introduction

Electronic digital computers became available in increasing numbers from the late 1940s onward and are today essential tools for research in most, arguably all, scientific fields (Haigh and Ceruzzi 2021, 29-54). In the 1940s and early 1950s computers were conceived primarily as machines for performing mathematical computations and so their diffusion in the natural sciences might seem to need little explanation: computers were the right tools for the job, one might assume, and scientists quickly understood it. However, the early days of scientific computing still await systematic exploration, and studies of specific scientific fields suggest that the introduction of computer-aided methods in research was not generally supported by the broader scientific community and was often due to local networks and personal initiatives.¹ Moreover, while computers have often been presented as agents of innovation, Jon Agar convincingly argued that, in many cases, they were not introduced in scientific fields in combination with new methods, but rather to perform tasks previously dealt with by other means (Agar 2006). Thus, to understand the early history of scientific computing, it is necessary to also take into account the diversity of scientific traditions, asking not only whether and how computing changed them, but also how the goals, methods and ideals of different fields shaped and appropriated computing. The working hypothesis of this paper is that the introduction of computing in various scientific fields from the 1950s onward resulted in the emergence of a variety of situated cultures of scientific computing such as those of computational statistical mechanics (Battimelli and Ciccotti 2018; Battimelli, Ciccotti, and Greco 2020), astrophysics (Anderl 2019), bioinformatics (Stevens 2013), or computational chemistry (Hocquet and Wieber 2017). These and other cultures thrived alongside each other, leading to a fragmentation which in part persists until today. In the present paper I investigate the emergence of a culture of scientific computing in high-energy physics by reconstructing the introduction

1 Examples are: Battimelli and Ciccotti (2018); Battimelli, Ciccotti, and Greco (2020); Borrelli (2019); de Chadarevian (2002); Mounier-Kuhn (2010); Park (2003); Varenne (2010). On the lack of systematic studies on the history of scientific computing see Agar (2006) and Borrelli and Wellmann (2019).

of computers at the European Organization for Nuclear Research, better known under the acronym CERN.²

CERN was founded in 1954 in Geneva (Switzerland) and purchased its first computer, a Mercury produced by the British firm Ferranti Ltd, in 1956. Thus, the developments analyzed here played out around the middle of the 1950s, a period of utmost importance and complexity for the history of computing in general, and of computing in the sciences in particular. This was a time in which both the technological aspects and the public image of computing were rapidly evolving, as computers became commercial products while at the same time finding their way into new areas of science and engineering. As an increasing number of scientists came into contact with the new machines, they formed their own images of computers and computing, and these were not necessarily shaped by those of computer experts. The beginnings of computing at CERN were part of this broader, rapidly evolving picture and contextualizing them is essential not only to understand the specific events discussed here, but also to appreciate both the fragmentation and the interconnection of different historical aspects of computing practices in science. High-energy physics is a particularly interesting case study, as in the early 1950s it was still an emerging discipline, rapidly becoming independent from nuclear physics, of which it had previously been a part. The historical constellation analyzed in this paper can thus be described as the encounter of two emerging knowledge cultures: computing on the one hand and high-energy physics on the other.

I will start by outlining the emergence of high-energy physics in the 1950s, its connection to nuclear physics and the creation of CERN (section 2), then I will discuss how in those years computing started to find its way into different scientific fields (section 3) and more specifically into nuclear and high-energy physics (section 4). Section 5 moves briefly away from the scientific context and provides an overview of the computer market around 1955, in order to familiarize the readers with the commercial offerings from which CERN could choose its new machine. After that, sections 6 to 10 relate the process leading from the first idea of buying a computer for CERN in the summer of 1955 to the decision to purchase a Ferranti Mercury in December of the same year. I conclude by arguing that the CERN case study supports the working hypothesis that the first forms of what we today regard as scientific computing only slowly emerged during the late 1950s, transformed from a rather marginal practice often performed by female staff to a core component of research—and from a highly risky commercial enterprise to a small, but potentially very profitable market sector. Yet scientific computing emerged not as a universal new

2 The acronym stands for “Conseil européen pour la recherche nucléaire” (European Council for Nuclear Research), the French name of the council tasked with planning the new laboratory.

method of research, but as a fragmented landscape of practices firmly bound to specific disciplinary networks of human expertise and digital resources. Indeed, I argue that CERN's culture of scientific computing was not shaped primarily by the needs of the laboratory, but rather by the way the computer was imagined as a new tool within the high-energy physics community.

2. Nuclear and High-Energy Physics in the 1950s and the Founding of CERN

High-energy physics studies the behavior of elementary particles at the highest experimentally accessible energies, and it emerged as a stand-alone field during the 1950s.³ Before that, it was largely regarded as a subdiscipline of nuclear physics, which had in turn emerged as a new area of research in the 1930s. The study of nuclear phenomena was intensively developed in the early 1940s in the context of nuclear weapon research, but after the war much of the knowledge gathered was declassified, and a number of new nuclear research laboratories were founded, pursuing not only military research but also the development of nuclear reactors for civil applications. The main ones in Europe were the British Atomic Energy Research Establishment (AERE) founded in 1946 at Harwell near Oxford, and the research center of the French Commission for Atomic Energy (Commissariat à l'Énergie Atomique [CEA]), founded in 1945 at Saclay near Paris (Amouyal 1990, Lawson 1997). As we shall see, these two laboratories played a very important role in our story, since their representatives shaped both the planning and realization of CERN and its computing culture.

In the early 1950s, while research on nuclear physics was expanding, experiments delivered increasing evidence of phenomena at a smaller, subnuclear scale, such as the production of new particles (pions or muons) or the unexpected behavior at very high energies of known components of matter (electron, protons, neutrons). Soon, increasingly powerful particle accelerators were built to explore how subnuclear objects reacted to collisions at higher and higher energies. This new field of enquiry went initially under the name of "high-energy nuclear physics," as opposed to nuclear physics proper, which dealt with nuclear fission, nuclear decay and other phenomena taking place at (relatively) lower energy. Initially, it was assumed that the same theory would be able to explain both higher and lower energy phenomena, but this turned out not to be the case, and by the end of the 1950s the fields had become separate disciplines: nuclear physics and high-energy physics, the latter often referred to as particle physics. This situation helps understand why in 1955

3 The following overview on nuclear and early high-energy physics is based on Brown and Hoddeson (1986); Brown, Dresden, and Hoddeson (1989); and Hughes (2002).

CERN physicists were dealing with high-energy phenomena yet at the same time were still part of the nuclear physics community.

Let us now look more closely at the Geneva laboratory.⁴ The idea of establishing a common European research center emerged in the aftermath of the Second World War as one of many initiatives to foster social, economic, and cultural cohesion among European countries. Moreover, thanks to the financial contribution of more than one nation, the center would be able to afford research infrastructures at the level of US institutions: an essential point for scientists wishing to build powerful particle accelerators. However, it would be wrong to think of CERN in 1955 as the huge research center it is today, whose technology is assumed to be always at the cutting-edge level. In discussing the early history of CERN, historian John Krige specifically tried to reconstruct how scientists and managers in 1954 imagined CERN would look in 1960, and his conclusion was: “The first striking feature of the prevailing mentality in the early 50s is that there was *no idea of intrinsic growth* of the basic nucleus of the laboratory” (Krige 1987b, 283, italics in the original). The initial plan was to invest heavily in constructing CERN (circa 120 million Swiss Francs [CHF] over 7 years) but only allow a much smaller budget for running it (ca. CHF8.6 million per year). In fact, Krige continues, the costs of the experimental program (not counting infrastructure) ended up amounting to CHF20 million for 1959 alone: “No one [in 1954] could have anticipated the qualitative transformation that high-energy physics was to undergo at the end of the decade” (Krige 1987b, 285). The rapid evolution and expansion of high-energy physics can be seen as in many ways similar to that of computing, and it might be speculated that this similarity played a role in making high-energy physics laboratories like CERN or the Stanford Linear Accelerator Laboratory (SLAC) important factors in the development of computing in general and scientific computing in particular (Borrelli 2019, 2022; Galison 1997).

In 1955, though, CERN was planned simply to host the first European large-scale particle accelerator, the Proton Synchrotron (PS), accelerating protons along a circular path. Since European expertise in designing and building accelerators was in the hands of physicists working at nuclear research facilities, especially Harwell and Saclay, these institutes participated in CERN funding negotiations and planning. Moreover, a number of CERN scientists were recruited from those laboratories. One of them was Lew Kowarski, who moved from Saclay to CERN in 1952 and, as we shall see, played a key role in the events to be reconstructed below. Another one was John Adams from Harwell, who in 1954 became the first director of the PS group at CERN, leading the planning and construction of the accelerator.

4 On the early history of CERN, see Hermann, Belloni, and Krige (1987–90), and especially Krige (1987a) and Pestre (1987), on which the following overview is largely based.

The CERN charter was signed in 1952 and the laboratory was inaugurated in June 1954, but initially no electronic computing services were foreseen. In a technical note from October 1953 the only “electronics” mentioned are circuits and memories to store detector output and, according to theorist Franz Cerulus, CERN’s computing facilities initially consisted of “two charming British ladies with mechanical desk calculators” (Beck 1964; Cerulus 2012; LK98.1⁵). Shortly after CERN’s inauguration in June 1954, two European computer manufacturers, the French Société d’Électronique et d’Automatisme (SEA) and the British Ferranti, heard from third parties that CERN might be interested in electronic computers and wrote offering their services (LK98.2, 3, 4). The answer was that CERN was not planning to buy an electronic computer (LK98.5, 6). One year later the situation had changed, and by the end of 1955 CERN would order its first computer, a Ferranti Mercury: What factors were behind this change of mind? Which historical actors, technological and industrial developments, and scientific networks played a role? Why was a Ferranti Mercury chosen? These are the questions that I will attempt to answer in the following pages.

3. Computing in the Sciences in the Early 1950s: Some General Remarks

Computing is today such a central component of the natural sciences that it may be difficult to imagine the situation in the 1950s, not simply as far as the technology available is concerned, but also with respect to the attitude towards computing among scientists. Therefore, this section offers a general outline of early developments in scientific computing, before the specific case of nuclear and high-energy physics is discussed in the next one. The section is written primarily for readers not familiar with the topic, but may also be of interest to historians of computing, as I here also anticipate some general conclusions of the paper.⁶

Today we regard computers in sciences as tools of research, but up to the early 1950s research with computers was often research about computers, aimed at improving their design, material technology and performance, and usually done on machines that the researchers had built themselves.⁷

- 5 References to documents found in CERN’s archive are here and in the following given in shortened form, with the full references listed in the reference section.
- 6 Due to the lack of systematic studies, the overview offered in this section is only an incomplete sketch to be revised and enriched by further investigations. It is based on: Agar (2006); Anderson (1986); Battimelli and Ciccotti (2018), Battimelli, Ciccotti, and Greco (2020); Bonfanti (2012); Borrelli (2019); de Chadarevian (2002); Galison (1997); Haigh and Ceruzzi (2021); MacKenzie (1991); Mounier-Kuhn (2010); Nash (1990); Seidel (1998); Stevens (2013); Varenne (2010); and Yood (2013).
- 7 Much research has been devoted to this topic. For an overview and references see for example Haigh and Ceruzzi (2021), Campbell-Kelly et al. (2023), and Haigh (2019), on

Computing was an important and rapidly expanding field, yet a still relatively small one, especially as far as European research institutions were concerned. It was only during the 1950s that computers became available in increasing numbers and variety, and could thus be employed within and beyond science and engineering. One might plausibly assume that scientists would have been more familiar with and interested in computers than business people or administration workers, yet this idea needs qualification. In the early 1950s most scientists only knew much about computers if they had been directly involved in their development, and there was no compelling reason why they should believe, or even speculate, that using the new machines might be helpful in their own work. In fact, many did not, and even some computer experts doubted that more than a handful of powerful machines would be needed in science (Haigh and Ceruzzi 2021, 22; Tweedale 1992, 100-101). Up to the early 1950s, even in computationally intensive areas of physics research manually operated tabletop machines were usually deemed sufficient for all practical purposes (Park 2003).

To better understand this skepticism one has to remember that computers at the time were expensive, not easily accessible, and often experimental machines. Even where a computer was available, off-the-shelf solutions rarely existed for the specific problems dealt with in a particular area of research. Computer experts, although in principle interested in promoting their products and results, had neither the resources nor the desire to provide full-time services to other scientists. Computer manufacturers, on the other hand, were hardly in a position to produce custom-built tools for scientists as they did for accounting or statistics, due to the variety and often rapid changes in scientific research. Cooperations between those firms and scientific institutions only took place when a strong state backing existed, as was the case for US nuclear weapon research, which in the 1950s prompted joint ventures of the two main such research centers, the Los Alamos and Lawrence Livermore National Laboratories, with IBM, Remington Rand and, later on, Control Data Cooperation (CDC) (MacKenzie 1991; Seidel 1998). Yet this was a singular, fortuitous case which led at least some US high-energy physicists to be early computer users (Seidel 1998, 32). In general, though, access to and interest in computers in the 1950s was rather limited, and so the diffusion of computing practices in the sciences was often linked to scientists who, having somehow come into contact with computing, had acquired both an interest in developing or promoting the development of computational methods for their discipline and the conviction that this endeavor would eventually result in a new, profitable research strategy. This was the case for the first computer simulations in molecular dynamics (Battimelli, Ciccotti, and Greco 2020), for the introduction of computational methods in genetics (de Chadarevian 2002) and in quantum

which this overview is based.

chemistry (Park 2003), for the first Monte Carlo event generators in particle physics (Borrelli 2019), and for computer-aided track reconstruction in bubble chambers (Galison 1997). The computational tools that emerged in this period were the result of diverse processes of more or less extensive appropriation, transformations, and at times even reinvention of methods developed in computing research.

When asking about the beginning of computing at CERN in 1955, it is therefore important to ask where interest in, and information about, the topic may have come from. A few conclusions may be anticipated here. In the first place, there was no pressing computational task at CERN that could only be performed with an electronic computer. In the second place, CERN's interest in computers was not the outcome of good marketing on the part of computer manufacturers. In the third place, there was at the time no overarching political strategy promoting scientific computing, as would be the case in the 1980s with the founding of supercomputing centers (Gugerli and Wichum 2021). As we shall see, an important source of interest in and information about computing at CERN and other research institutions was contact with academic computer experts working to promote their creations.

In the late 1940s and early 1950s electronic digital computers were built in research institutions mainly to explore and develop the potential of the new computing methods, but they could also be made available locally for colleagues to use, to raise interest in the machines and justify their costs, or simply due to personal contacts. These computers and the groups building and managing them often constituted the first contact of scientists with the new machines and so played a key role in the emergence of cultures of scientific computing. In the USA, where defense funding for these initiatives was available, some research centers built and upgraded their own computers throughout the 1950s. Examples were the MANIAC (Mathematical Analyzer Numerical Integrator and Automatic Computer) series at Los Alamos in New Mexico (Anderson 1986) and the AVIDAC (Argonne Version of the Institute's Digital Automatic Computer) at the Argonne National Laboratories in Illinois (Yood 2013), both of which, as we shall see, were also used as research tools in nuclear and high-energy physics.⁸ Universities with extensive defense funding could also participate in joint ventures with large computing firms like IBM and Remington Rand (Lewis 2017; MacKenzie 1991; Seidel 1998). In Europe, too, computers were built in universities and research centers. This was the case early on in the UK, where researchers working at the National Physical Laboratories (NPL) in Teddington near London developed the ACE (Automatic

8 Los Alamos and Argonne were founded as sites for the Manhattan Project and continued receiving state funding after the war, which they partly used for developing computers with a view to employing their computational power in the development of defense technology.

Computing Engine) during the Second World War, while in the later 1940s the Mark I was built at Manchester University and the EDSAC (Electronic Delay Storage Automatic Computer) at Cambridge University.

For our topic, a particularly important figure was Maurice Wilkes, leader of the Cambridge Mathematical Laboratory where the EDSAC had been built (Ahmed 2013, 56, 62-62, 94-97; Campbell-Kelly 2019; de Chadarevian 2002, 107-25). In the early 1950s Wilkes' institute became a hub for diffusing knowledge about computers both locally and among the broader scientific community. Starting around 1950, Wilkes regularly organized courses and summer schools in coding and conferences on scientific uses of computing, and in 1951 published one of the first textbooks of programming, also used for the lectures given at the computing schools held in Cambridge (Wilkes, Wheeler, and Gill 1951). Besides promoting education and information exchange, Wilkes also made European scientists familiar with the so-called open shop model of computing resource organization, in which scientists wishing to use computers were encouraged to learn to code and, to a certain extent, directly use the computer, instead of delegating programming and operation to mathematicians and technical staff (Haigh and Ceruzzi 2021, 109). As I shall discuss later on, this mode of working particularly suited experimental physicists, who traditionally preferred to have close control over their research instruments.

4. Computing in Nuclear and High-Energy Physics in the Early 1950s

The interplay between the development of computing and research in nuclear physics has been studied in most detail for the case of the two main US nuclear weapon research laboratories, Los Alamos and Livermore (Lewis 2017; MacKenzie 1991; Seidel 1998). In those cases, from the early 1950s onward the financial backing of the US Atomic Energy Commission (AEC) offered computer manufacturers a chance to experiment in developing new computer hardware and architecture with a relatively low financial risk and with the aim of using the eventual results to expand their line of commercial products. Put in these simple terms, the development might be interpreted as resulting from a rather straightforward synergy: physicists wanting more computing power, politicians looking for increasingly powerful weapons during the Cold War, and computer manufacturers striving for profit. However, detailed historical studies have shown how the developments were all but straightforward: on the one hand, big computing firms, notably IBM, did not always have an interest in such cooperations, and often did so mainly to keep up with their competitors; on the other hand, physicists had at times diverging visions of what desirable features a computer should display. Without going

into the details of these analyses, it is worth mentioning a result by Nicholas Lewis (2017), who analyzed in detail the purchase strategy for computers at Los Alamos between the late 1950s and the early 1970s. Lewis highlighted the diversity of factors playing a role in the decisions and in particular noted the growing tensions among Los Alamos scientists from two different emerging cultures of computing: “computing as a science and computing as a service” (Lewis 2017, 36).

While the relationship between computing and nuclear research, both for civil and military uses, in the USA has received much attention, few studies have been devoted to European cases, and the following remarks are largely based on recollections of historical actors (Amouyal 1990; Howlett 1979; Mounier-Kuhn 1998; Raymond 1989; Tweedale 1992). In this context, Wilkes’ seminars in Cambridge are important, because through them scientists working at the nuclear physics centers of Harwell and Saclay learned about computing. Let us first look at Saclay. In general, French scientists became interested in computing only relatively late, but the CEA in Saclay was among the first French institutions to acquire computers (Amouyal 1990; Mounier-Kuhn 2010, 212–13). Albert Amouyal, at the time director of its Computing Department, recalls how in 1954 he took the initiative of purchasing a computer, a topic he had become familiar with by taking part in Wilkes’ courses in Cambridge (Amouyal 1990, 220). In 1955 the CEA finally ordered a computer: a Ferranti Mercury. According to Amouyal’s recollection, there was originally no single specific task for which the computer was needed, but it soon became a useful tool. Indeed, in those years new numerical methods for the study of nuclear reactions were being developed, and so it is not surprising that interest in computing spread among nuclear physics laboratories (Birkhoff 1990). Yet purchasing a computer was still not an obvious step. This was the case also for Harwell, about which we are informed by the recollections of Jack Howlett, head of its computing division (Howlett 1979). Initially, computing services at Harwell consisted of a number of operators, mostly women, using mechanical or electro-mechanical machines, but eventually the need to perform heavier computations led some scientists to take an interest in digital electronic computers. In 1950 Howlett took part in “what must have been one of the first programming courses in Britain,” organized in Cambridge in cooperation with Wilkes’ Mathematical Laboratory (Howlett 1979, 5). In 1951 Harwell’s Electronic Department delivered a custom-built computing machine, the so-called Dekatron Computer, and in 1953 two experimental physicists from Harwell, John Adams and Mervyn Hine, used the ACE computer at NPL in London to perform computations for the planning of the CERN PS machine. Both Adams and Hine moved to CERN in 1954, Adams as director of the PS Division (Hermann 1987, 546, 554–55; Lawson 1997, 22). Eventually, the increasing computational load led Harwell physicists to turn towards commercial computers. In 1954 they used the Ferranti Mark 1* installed at the Atomic Weapon

Research Establishment (AWRE) in Aldermaston and in 1956 Harwell bought a Ferranti Mercury. Howlett notes that this purchase met “some opposition from, curiously enough, parts of the scientific population: not everyone was convinced that a computer was really a necessity” (Howlett 1979, 8-9). This is one more piece of evidence of the negative or indifferent attitudes of many (nuclear) physicists towards computing machines and that their introduction into research practices was not an obvious choice.

It is interesting to note that, although Adams had used computers for the planning of the PS accelerator, in 1954 he apparently expressed no interest in CERN's acquisition of a computer, though once Kowarski had brought up the idea, he was ready to support it. Besides once more pointing out the general lack of interest in computing in the high-energy physics community, this episode also indicates that computations for the planning of the PS were not the initial motivation for CERN's purchase of a computer. Neither were Monte Carlo computations, which in 1955 were widely used in nuclear physics, but had yet to find application in high-energy physics (Borrelli 2019). In general, numerical tools developed for nuclear physics were of little use for the study of subnuclear particles, due to the difference of phenomena investigated. Therefore, although interest for computing at CERN was partly due to its increasing importance at nuclear laboratories like Harwell and Saclay, its concrete uses in Geneva were still largely to be defined.

What about computational tools developed specifically for particle physics? At the time, the only one that was discussed at CERN was the computer-aided method of data analysis developed in the early 1950s by Enrico Fermi (Chicago University) in collaboration with computer expert Nicholas Metropolis for the MANIAC at Los Alamos, where Fermi had worked in the context of the Manhattan Project (Anderson 1986). After Fermi's death in 1954, his colleague Herbert Anderson, a Chicago particle experimentalist, expanded the method further, implementing it on the AVIDAC (Anderson et al. 1955; Anderson 1986). This was probably the earliest use of computers for data analysis in what were at the time high-energy experiments and, as we shall see, Anderson was one of the experts contacted by Kowarski when looking for information on computing.

However, the phenomena studied at CERN were different from those for which Fermi and Anderson employed their method, so no direct knowledge transfer was possible: CERN scientists' decision to consult primarily nuclear and particle researchers on whether to acquire a computer was less obvious than it seems. Beside physicists as Harwell, Saclay and Chicago, other computer experts were in principle available to CERN scientists, but they were not consulted. For example, in nearby Zurich there was the group led by Eduard Stiefel at ETH (Eidgenössische Technische Hochschule), where during 1952–1955 the ERMETH (Electronic Calculating Machine of the ETH) computer

had been built (Speiser 2004). In Germany, custom-built computers were developed in the early 1950s at the University of Darmstadt and at the Max-Planck-Institute (MPI) of Physics in Göttingen (Biener 1999; Eyßell 2010). The Göttingen machine was employed for computations based on Werner Heisenberg's model of particle interactions, but, although Heisenberg in 1955 was the president of CERN's Scientific Policy Committee (SPC), computer experts in Göttingen were not contacted.⁹ In Italy, in the early 1950s commercial computers had been acquired by research institutes in Rome (Ferranti Mark 1*) and Milan (CRC 102-A), while at Pisa University an original machine, the CEP (Calcolatrice Elettronica Pisana), was being built, motivated by a suggestion made in 1954 by Enrico Fermi to Gilberto Bernardini and other particle physicists (Bonfanti 2012). Bernardini had worked with computers at the University of Illinois and in 1955 was collaborating with CERN, but never proposed purchasing a computer, although he supported Kowarski's plan (CERN Courier 1962). No Italian computer experts were contacted by CERN in 1955.

5. The Computing Market of 1955

Electronic, digital computers started becoming commercial products at the beginning of the 1950s, and at the time it was far from obvious that they would ever be successful as such.¹⁰ Would there be enough demand for them in military and civil research? If not, could machines created in state-funded defense projects be adapted to perform business tasks like accounting or inventories, and at a price acceptable to commercial firms? Despite these uncertain prospects, around 1950 the US start-up UNIVAC showed that interest in computers could be raised among businesses as well as state-backed research and administrative institutions. At that point larger companies, too, entered the market, most of them manufacturers of office machines or electronic apparatuses. This happened first in the USA and UK, where the relevant know-how existed, but in the early 1950s French, German and Italian companies also entered the market. By 1955 various European computing firms were active and striving to expand, and it was only during the 1960s that they lost ground to US competitors (Mori 2019).

Entering the computer market was not an easy decision for a firm: in the 1950s, computing was a rapidly developing field, both in the material technology of circuits and memories, and in what would later be referred to as architecture

9 See Heisenberg ([1958] 2001). I am grateful to Alexander Blum for sharing with me his knowledge of Heisenberg's research activities and pointing out relevant sources.

10 For recent overviews of the history of computer manufacturers, see Campbell-Kelly et al. (2023), Cortada (2012), Gandy (2013), and Haigh and Ceruzzi (2021), on which the following passages are based. For an overview of the economical, social and political context of the rapid development of computing manufacturing in the USA during the 1950s see Aspray and Loughnane (2019).

and software. Due to the experimental technology, developing computers was a substantial financial risk, and in the early 1950s computer manufacturers aimed for a balance between innovation, reliability and affordable price, especially since business clients often settled for reliable and affordable, if somewhat slower, machines based on (slightly) older technology. As noted above, only a few larger, state-backed firms allowed themselves to invest in the development of highly experimental machines with great speed, large memory, and other new features whose payoff was not clearly predictable. This development can be described in retrospect as a differentiation between computers for business and computers for science, but, as we shall see, in fact many machines produced in the early 1950s could be and were employed to both aims. In a few words, we can say that the process in which computers during the early 1950s became commercial products had two sides. On the one hand, manufacturers explored market chances, adapting their offering to them, while on the other business firms and research institutes learned about the new machines and their potential uses. By 1955, many computer firms had started focusing on the business market, whose needs and expectations were better known and more standardized than those of the varied and often rapidly changing scientific users. From this background one can conclude that when CERN decided to purchase a computer there was no obvious candidate. In the relevant CERN documents, a number of manufacturers from Europe and the USA appear, and in the final part of this section I provide some information on them, with a few words anticipating their contacts with CERN, which will be discussed in more detail in the second part of the paper.

Among US firms, only two were mentioned regularly in CERN documents—UNIVAC and IBM—both of them praised as offering very good, yet too expensive products. UNIVAC belonged in 1955 to the firm Remington Rand and produced the UNIVAC 1, one of the most powerful machines of the time, which was sold to civil and military research institutes, commercial firms, and state administration agencies (Campbell-Kelly et al. 2023, 110-22; Haigh and Ceruzzi 2021, 22-27). The office machine manufacturer IBM around 1955 offered two powerful, but expensive machines, the IBM 701 (1952) and 704 (1954), and a more affordable, but less powerful one, the IBM 650 (1953) (Campbell-Kelly et al. 2023, 123-27; Gandy 2013, 213-64; Haigh and Ceruzzi 2021, 55-62). These computers could be used both for scientific and for business and administrative tasks, but the success of the IBM 650, which sold more than 1,000 units, was due to business firms. IBM's greatest strengths were its marketing and services, and the firm had started to expand in Europe already in the late 1940s. In 1955, it actively sought contact with CERN to promote its products.

Although CERN reports praised US computers, it is quite clear that the center preferred to purchase European products. Accordingly, four British and two French manufacturers were taken in more or less serious consideration and

eventually a British firm won the contract: Ferranti Ltd. As we saw above, Ferranti at the time also delivered machines to Harwell and Saclay. Although this fact represented an important motivation for CERN to purchase a Mercury, it would be wrong to assume that the Geneva laboratory was passively following Harwell's and Saclay's lead. As I shall argue in more detail later on, the CERN community conformed to the choice made by the nuclear physics laboratories because they shared the same ideal of what scientific computing in nuclear and particle physics should look like.

Let us now take a closer look at the British and French manufacturers whose products were discussed in CERN documents, starting with the one which eventually won the bid: Ferranti. Ferranti Ltd was a UK electronics firm which had participated in the development of radar during the Second World War (Gandy 2013, 123-62; Tweedale 1992). In the late 1940s the firm decided to enter the computing market, and joined forces with scientists at Manchester University who had recently built the electronic computer Mark 1. With support from the British government, Ferranti set up a computer group and in 1951 started promoting the Mark I and its upgraded version Mark I*, eventually selling nine machines between 1951 and 1957, mainly for research work. The Mark I and I* were very powerful, fast machines, but quite expensive. At that time the Ferranti management and the British government realized that a more affordable machine would fare better on the market and this idea led to the production of the Ferranti Pegasus. From 1956 onward more than 35 Pegasuses were delivered, to be used both for research and for administrative work. At the same time, though, Ferranti developers cooperating with engineers at Manchester University were pursuing another project since 1952: the Mark II, a computer that should have greater speed and a larger, faster memory than the Mark I* or the Pegasus. The new product would also implement an innovative feature, namely floating point number storage.¹¹ From 1955 onward, this machine was advertised and sold as the Ferranti Mercury, though deliveries, initially planned for 1956/57, were delayed until 1957/58. The Mercury was used primarily for research work, especially by nuclear physics laboratories like Harwell and Saclay, and right from the beginning was praised in CERN reports as a fitting machine for the laboratory.

While Ferranti was at least partly involved in developing increasingly powerful, but expensive, computers, other British manufacturers mainly focused on less powerful but more affordable machines, probably spurred by the commercial success of the IBM 650. One of them was Elliott Brothers, a British instrument

11 Floating point is a way of digitally storing numbers which was introduced around that time, and is today standard. It employs the exponential notation used in science, where a number is (approximately) represented by its significant digits multiplied by an exponent of 10, e.g. 2.6×10^6 for 2,600,000 (Haigh and Ceruzzi 2021, 32-33). On the relevance of floating point storage and operations for increasing computational speed, see MacKenzie (1991).

manufacturer that had been a defense contractor for radar development during the Second World War (Lavington 2011). Around 1950 the British government agency National Research Development Corporation (NRDC) joined forces with Elliott with the aim of commercializing national developments in computing. Around 1950 two projects were envisioned: a computer primarily aimed at business clients and a largely non-described “advanced computer.” The first machine was designed, produced, and from 1953 onward successfully marketed as the Elliott 401, while the “advanced computer” was never built. One of the designers of the Elliott 401, Bill Elliott, eventually moved to Ferranti, where he helped realize the Pegasus.¹²

A second British firm whose products were considered by CERN was the English Electric Company (EE), which had become involved in computer development in the late 1940s as partner to a state-funded project at the above-mentioned NPL in London, with the aim of realizing a commercial version of the ACE computer (Gandy 2013, 189-210). Starting in 1955 this machine was marketed as the Deuce, and by 1961 20 units had been sold to military and civil state institutions, aircraft manufacturers, universities and atomic power agencies, to be used for both research and administrative work. In principle, the Deuce was a serious candidate as a CERN computer, since in 1955 the machine was new but already successfully tested. However, CERN scientists expressed the opinion that the Deuce was not powerful enough. As we shall see, it is not obvious why that was the case. Finally, one of the CERN documents related to the computer purchase mentions the LEO I, but dismisses it quickly as not fitting. This is interesting since the British firm LEO Computers had strong ties to Wilkes’ group, which, as we saw in the previous section, had a central role in promoting computing among physicists. LEO Computers Ltd, a branch of the food manufacturer and catering firm J. Lyons & Co., had entered the computer market in the late 1940s by joining forces with Cambridge computer experts, as discussed in detail in Elisabetta Mori’s contribution to this volume (see also Mori 2022). The EDSAC computer, built by Wilkes’ group, served as a prototype for the LEO I, yet apparently the Cambridge institute had no influence on the choices of Harwell, Saclay or CERN about which computer to acquire.

Beside British manufacturers, two French firms bid for the CERN contract. The more successful one was the Société d’Électronique et d’Automatisme (SEA), founded in 1948 and specializing in industrial automation devices, analog computers, and flight simulators (Mounier-Kuhn 1998, 11-16; Raymond 1989). SEA’s main clients were French Aeronautics, the French Navy and deciphering services, but in the early 1950s the firm moved into the production of electronic digital computers, and to this end its engineers went to Cambridge to attend Wilkes’ summer schools (Raymond 1989, 268). The result was a new

12 Bill Elliott had no family connection with the firm Elliott Brothers (Lavington 2011, 38).

series of “Calculatrices Arithmétiques Binaires (CAB),” which included the CAB 2022, delivered in 1955 to French Aeronautics. In 1956 SEA started production of the CAB 3000 line, which included the CAB 3040, a machine planned but never actually built. On paper, the CAB 3040 possessed essentially the same features as the Ferranti Mercury and, as we shall see, was shortlisted as a possible CERN computer. In the end, though, it lost out to the Ferranti Mercury and, possibly because of this, was never built. The second French firm approaching CERN was the Compagnie des Machines Bull, originally established in the 1930s as a producer of mechanical, later electro-mechanical, office machineries (Mounier-Kuhn 1989, 2003). In the early 1950s Bull hired a number of engineers familiar with new developments in electronics and entered the computing market with the Gamma 3 (1952), which sold more than 1,000 units. Later on, the Gamma 3 was provided with a drum memory and became Gamma Tambour, which Bull promoted as their first stored program computer. As we shall see, Bull representatives actively sought contact with the Geneva laboratory through political channels, but the Gamma computers were never seriously taken into consideration at CERN.

6. Lew Kowarski and His Role in the Introduction of Computers at CERN

In 1955 CERN was organized into six Divisions, each with its own director, and the General Director of the laboratory was Cornelis Jan Bakker. Additionally, there were two external bodies: the Scientific Policy Committee, advising CERN on scientific matters; and the Financial Committee, which represented the nations financing it. In the process of computer purchasing a key player was a physicist named Lew Kowarski, at the time head of the Scientific and Technical Services (STS) Division and thus responsible for managing the center’s infrastructures. I will therefore begin with a short outline of his career up to the time he joined CERN.¹³

Born in Saint Petersburg in 1907, Kowarski fled Russia with his family in 1917, moving first to Vilnius (at the time in Poland), then to Belgium, and finally to France. Despite financial, bureaucratic and cultural difficulties he managed not only to complete his university studies in Lyon and Paris, but also to join Irene and Frédéric Joliot-Curie’s research group on nuclear fission. When Germany invaded France in 1940, he fled with colleagues to Great Britain with the group’s supply of heavy water. During the war, Kowarski worked in nuclear research first in Cambridge and later in Montreal, eventually coming back to France and becoming in 1946 director of the Department of Physics and Technology of the CEA in Saclay, where he supervised the design and

13 The following overview of Kowarski’s life and work until 1955 is based on Kowarski (1971) and Hermann (1987, 556).

construction of the first French nuclear reactor. Kowarski was adviser to the French delegation to the UN Commission on atomic energy control, as well as for the consultations to establish CERN. In 1952, when the provisional CERN organization was divided into four "study groups," Kowarski was appointed director of the laboratory group, responsible for organizing buildings, infrastructure and administration in the interim phase before the official CERN start in 1954 (Pestre 1987, 161). In connection with CERN's inauguration in 1954 there was a reorganization and Kowarski's laboratory group was split up into the Scientific and Technical Services (STS) Division under Kowarski, and the Sites and Building (SB) Division and the Administration (ADM) (Hermann, Belloni, and Krige 1987–1990, vol. 2, 826).¹⁴ Clearly, this step meant a reduction of Kowarski's responsibilities, a topic he dealt with in detail in an oral history interview in 1971. The subject is of relevance for our theme, and I will summarize it here.

According to Kowarski, his appointment to CERN in 1952, though fitting his experience and management skills, was primarily due to a restructuring of the CEA that involved his departure and, as he put it, "certain deficiencies often had to be observed, and it would have been far easier to say that 'Kowarski left an important position in Paris to occupy an important position in Geneva'" (Kowarski 1971, Session VII). In other words, Kowarski felt he had been promoted away from the CEA to a position for which some of his CERN collaborators might think he was not fully qualified, not being a high-energy physicist, and that this situation had eventually resulted in a slow curtailing of his areas of influence. Whether or not Kowarski's interpretation was correct, the situation caused him so much stress that in early 1955 he had to take a break for health reasons. When he came back in the summer of 1955 he worked part-time and tried to make the best of his situation. It was in this context that he took an interest in computers:

I began to take it easy and look at what specified task I could occupy my troops with ... Two of them were found immediately. One was developing the production of liquid hydrogen on the site. ... The other was the terribly newfangled subject of electronic computing machines, later on to be known by the shorter name of computers. (Kowarski 1971, Session VII)

Asked why he had chosen computers as a topic to focus on, Kowarski stated:

For many reasons I was already interested in computers, as shall we say a great factor in industrial civilization, and in fact I started getting my first information on the subject already early in '55. By summer '55, at some meeting when I said something, "Well, CERN probably should have

14 The three other CERN Divisions were one each for the two planned accelerators, the small Synchrocyclotron (SC) and the main machine, the Proton Synchrotron (PS), and a theoretical group (TH).

a computer, let me look into it," everybody was quite happy to let me play with it, and so I was left to prepare plans for a computer for CERN. (Kowarski 1971, Session VII)

From this statement it might seem that one single man, Kowarski, was responsible for CERN's decision to acquire a computer. However, Kowarski's interest in computers was not a personal whim, but reflected rapid transformations in the techno-scientific landscape of his time which, as we saw above, were already underway in the USA and just starting in Europe. Most research scientists were not aware of or interested in these changes, but an experienced science manager like Kowarski saw the growing relevance of the new machines for big science.

7. The Developments at CERN: August–December 1955

7.1 The First CERN Meeting on Computing (August 1, 1955)

Kowarski recalled that in the summer of 1955 he proposed that CERN should purchase a computer. In a "draft note" of a report to the Director General written between August 2 and 8 he wrote:

Following the recommendation of the CERN Committee of July 14th, I called on August 1st a meeting, comprising representatives of [all CERN Divisions], to discuss the possibility of acquiring an electronic computer. Mr. Adams was able to secure the presence of Dr. Howlett of Harwell ... An additional conversation took place on August 2nd between Dr. Howlett and most of those who attended previously.¹⁵

I was not able to find any further information on the recommendation from July 14 mentioned by Kowarski, but we may assume that it was related to Kowarski's mentioning the idea of a CERN computer. The only computer expert at the meeting in August was apparently Howlett, head of computing at Harwell, which was at the time in the process of acquiring a Ferranti Mercury. CERN directors were seeking information on computing from a trusted peer: a scientist working with computers at a nuclear physics laboratory connected to CERN. We also recall that Adams, now director of CERN's PS Division, had had some experience of computing in 1953, though he did not seem to be active in the field later. The decisions reached on August 1 and 2 were that at least CHF500,000 should be allocated for the purchase, and that "[i]n this price range, the Ferranti Pegasus is the best buy available at present," while

15 The note LK1.1 is undated, but it was written after August 2, 1955, as it reported the meeting held on that day, and before August 9, 1955, when the final version of the report was discussed in a directors' meeting (CERN/SPC/13).

"[a]nother Ferranti machine ('Mark II') is the best buy at the CHF1,000,000 level" (LK1.1). The Mark II was presented as clearly superior to the Pegasus:

Mark II offers a far wider fast memory. According to some preliminary enquiries (which will have to be checked) the absence of a comprehensive fast memory in Pegasus slows down the calculations of a certain type (of interest to the PS designers) so as to seriously impair its usefulness.
(LK1.1, 1)

The "Mark II" mentioned here is the machine that would be later called Mercury. In the note no computer manufacturer other than Ferranti is mentioned, and the difference between the medium-small Pegasus and the large Mark II/Mercury is underscored, leaving no doubt that the latter was considered much more fitting for CERN. Probably the outcome of this first meeting was shaped by Howlett, who had already formed a favorable opinion of the Mark II/Mercury, despite uncertainties about actual performance and delivery date. The decisions of the committee were stated in the report in these words:

- (a) A programming expert from Ferranti is to be invited to Geneva early in October, for a discussion of the suitability of either machine.
- (b) PS [the Proton Synchrotron Division] is to compile a selection of computing problems concerning the PS design; this selection would be sent to Ferranti, for their meditation, several weeks before the meeting.
- (c) Prof. Anderson of Chicago should be asked to supply examples of computing problems encountered in research carried out with a synchro-cyclotron, and to express his opinion on the importance of a comprehensive fast memory.
- (d) No recommendation as to the purchase of a definite type should be made before October. (LK1.1, 1-2)

In conclusion, another trusted peer, Anderson, should be asked for his opinion and a face-to-face meeting with Ferranti programmers should take place. The fact that both the PS Division and Anderson were expected to provide examples of relevant computations supports the hypothesis that no specific aim for the computer was envisaged at the time the notes were written.

The meeting with Ferranti representatives took place in early October, while particle and nuclear scientists with computer experience were contacted (LK1.2). Kowarski did not approach any other computer manufacturers, but various firms wrote to CERN, as word had spread that the laboratory was interested in buying a computer. As noted above, there seem to have been no attempts to seek advice from scientists involved in computer development in Zürich, Darmstadt, Göttingen, Pisa, or London. In November 1955 Kowarski

was contacted by Stiefel from ETH, who in a letter offered his advice for CERN's purchase of a computer (LK1.3). No answer by Kowarski is extant, so it seems that he did not take up Stiefel's offer.

7.2 Letter from Herbert Anderson (University of Chicago)

In early August 1955, Yves Goldschmidt-Clermont, a physicist in Kowarski's group, contacted Anderson, asking about his experience in computing in high-energy physics at Chicago University. Anderson replied on August 11, explaining that computers could be a great asset for research, but physicists would only be willing to engage with them if they felt they had a powerful tool in their hands:

In our own efforts to procure a computer for the University of Chicago, I have consistently held out against the smaller, slower machine in the hope that we would, in the end, be able to get a larger one. My principal objection to the smaller machine is not only that it is almost 10 times slower but that because of its limited capacity, it takes more ingenuity to code any but the most stereotyped problems. For a physicist, a certain potential barrier has to be overcome before he is willing to take the trouble and the time to learn how to use a machine. The more powerful the machine, the more worthwhile the effort becomes. The smaller machines, in my opinion, are just not powerful enough and fast enough to attract as many users as would be desirable.

The main point is that physicists will not make use of the computers until it becomes clear to them that a rather small effort made to learn how to use the machine will be very quickly repaid in terms of the added power which they will then have in solving problems in which they are interested. (LK1.4)

Anderson described experiences with students working on analysis of data from particle experiments: when he convinced them to code their problems on the AVIDAC instead of solving them by hand, they quickly realized that it was worth investing some time and effort in preparing a code which they could later use again and again:

They [i.e. students] feel that with the computer they have an advantage which their predecessors did not have. They develop the courage to go ahead and solve problems which would have seemed too difficult for them to do otherwise. (LK1.4)

This image of computers as making difficult tasks easy and even the impossible possible would shape their future reception in the sciences, independently of whether it corresponded to actual developments (Agar 2006). From Anderson's description we may also imagine that, while

researchers of his generation were perhaps not very interested in the new machines, younger scientists viewed them as a research tool with potential. In particular, computers promised to allow them to save a most valuable resource in scientific practice: time. Speaking of a student, John Simpson, solving a problem with a computer for the first time, Anderson wrote:

It is this enthusiasm for carrying out the more difficult kinds of computations that makes it worthwhile to have a machine. But if this is what is wanted, then it is important to have a versatile machine of large capacity, operating at high speed. Simpson's problem, which took several days of computation using the AVIDAC, might conceivably have been done in about a month using a drum-type machine. Under these circumstances, Simpson might have hesitated to invest a month of his time on this problem. If he did tie the machine up for a month, somebody else would have had to wait a month before he could get to the machine. (LK1.4)

Finally, Anderson described how the work with computers in his university was organized according to the open shop model, where "those who would like to make use of the machine, code their own problems and punch their own tape" (LK1.4). Control over their own research instruments was a feature which physicists appreciated.

7.3 Kowarski's Notes on a Report by Albert Amouyal (CEA Saclay)

Also in the summer, Kowarski wrote by hand two pages of notes in French summarizing a report written on June 2, 1955 by Albert Amouyal from Saclay in the context of the acquisition of a computer for his own agency.¹⁶ The notes are undated, but it is probable that they were written after the meeting at CERN in August, since they refer to the "Ferranti Mercury" and no longer to the "Ferranti Mark II," the provisional name of the machine.

Amouyal's conclusions as noted down by Kowarski were very similar to those reached in the CERN August meeting at which Howlett and Adams were present, suggesting that Saclay and Harwell scientists already shared a budding culture of scientific computing in nuclear physics. Kowarski summarizes Amouyal's position as: "In favour of Mercury: floating point, very flexible code, fast, memories, Harwell" ("Favorable à Mercury: virgule flottante, code très souple, rapide, mémoires, Harwell" (emphasis in the original)). "Harwell" clearly refers to the fact that the British atomic energy laboratory, too, was planning to purchase a Mercury. The next comments read: "Reliability not proven. Optimistic forecasts" ("Fidélité non-prouvée. Prévisions optimistes"),

¹⁶ Document LK1.5, from which all following quotes are taken. The original report by Amouyal is not among Kowarski's papers, and I was so far not able to trace it. Given his close contact with Saclay, Kowarski may have been allowed to see a copy of it and take notes.

followed by price estimates for the machine, its maintenance, and staff. Notes on the report's attachment follow: a summary of the decision criteria and the significant remark: "proven reliability = outdated machine" ("Fidélité éprouvée = machine démodée"). Once again, the fact that a computer was an experimental prototype was not seen as a fault, but rather as an indication that the machine was at the forefront of innovation.

A commented list of possible machines followed. The manufacturer Bull was dismissed as "expensive, no experience" ("cher, pas d'expérience"), the IBM 650 as outdated, and the IBM 704 as "excellent, much too important" ("excellente, bcp [beaucoup] trop important"). The CAB series by Société d'Électronique et d'Automatisme (SEA) was described as "= EDSAC" and, indeed, the engineers at SEA had acquired their expertise on electronic digital computers at Wilkes' Cambridge laboratory, where the EDSAC had been built. Moreover, the SEA libraries were "modest." The Elliott 402 had good reliability, but no fast memory and a too short word length, requiring "optimum programming" ("programmation optimum") to attain good speed. Kowarski concluded "we have to aim higher" ("faut viser plus haut").

The Deuce by EE was worth more consideration, as it was a very new machine successfully used for scientific computation: "first Deuce delivered May '55 to the NPL [National Physical Laboratory]. Very reliable. Very fast. ... Good fast and slow memories. Good library" ("1er deuce livrée Mai 55 à NPL. Très fidèle. Très rapide. ... Bonne memoires rap. & lente. Bonne bibliothèque"). However, it was not perfect: "very serious inconvenience: complexity of the code" ("inconvenient très grave: complexité du code"). Finally, the two Ferranti machines in development were described: the Pegasus was labeled as an "Elliott 401 ferrantized (NRDC)" ("Elliott 401 ferrantisée [NRDC]"), meaning a version of the Elliott 401 developed by Ferranti with the help of the British government agency the NRDC. It was deemed reliable, with a simple and flexible code and a very good library, but insufficient fast memory. Moreover, like all previous machines except the IBM 704, it had no floating point function, a feature instead available in the Mercury. Ferranti, according to the notes, had "the most experienced team in Europe" ("Equipe la plus expérimentée en Europe"), the Mercury's memory was good and about its libraries Kowarski wrote "not important in the beginning?" ("Bibliothèque pas important au départ?"). Kowarski quoted from the report: "a really different class than all others, except for the 704" ("Classe vraiment différente de toutes autres sauf 704"). This last sentence is taken literally from Amouyal's text, as indicated by the fact that Kowarski puts it in quotes, yet he clearly endorses the Frenchman's opinion, and I see this fact as supporting the hypothesis that at this time physicists interested in computers were trying to draw a distinction between scientific and business machines, with the former more powerful and cutting-edge than the latter, and thus more suitable for (large)

scientific enterprises. Whether this view corresponded to the actual situation is not obvious. For example, the Deuce, the IBM 650, the Elliott 401 and the UNIVAC were used both in science and business, and commercial clients often preferred smaller machines simply because it paid out to purchase a cheaper computer and allow a bit longer time for a given task. As seen in Anderson's letter, instead, for physicists the requirement that results should be obtained as fast as possible trumped considerations of price, and in the case of the Mercury also of availability and stability of performance. Here one may wonder how far the emphasis on fast results might rather be due to the competitive environment of high-energy and nuclear physics than to the cost in terms of wasted research time.

7.4. Contacts with Computer Manufacturers

A few days after visiting CERN in early October 1955, a Ferranti representative wrote to Kowarski that Harwell had ordered a Mercury, and that he looked forward to meeting the scientist in December in Cambridge at a seminar organized by Wilkes on the "Application of modern methods of computation in physical research" (LK1.2, 1.6). On November 11, Ferranti informed Kowarski that they had now drawn a production schedule and would be able to deliver a Mercury to CERN in April 1957, but only if CERN placed the order "in the next few weeks" (LK1.6).

While Ferranti enjoyed close contacts with CERN, other computer manufacturers strove to draw attention to their products. Already in late August representatives from EE had written to Kowarski: they had met "Heine" (probably Marvyn Hine) at the Atomic Energy Convention in Geneva and learned from him that Kowarski, Hine and Adams would soon be in London (LK1.7). They offered to demonstrate the Deuce at EE's London headquarters in September, an invitation which Kowarski declined (LK1.8). EE also contacted Adams, but Adams passed on all further correspondence to Kowarski, who rather bluntly told EE that CERN required a computer with fast memory, automatic floating point and great word length, asking if the firm planned to build one (LK1.9, 1.10, 1.11). Kowarski also offered to meet in December at Wilkes' conference, to which EE replied that they were looking forward to presenting their products (LK1.12).

The French firm Bull approached CERN through political channels. On September 26 Lyonel Wurmser, inspector general honorary at the French Ministry of Finance and Economic Affairs, wrote to CERN's "head of purchases," explaining that he had spoken to the director of CERN administration about the candidature of Bull for a contract for a computer, and enclosing a technical report on the Gamma Tambour (LK1.13). No reply from CERN is found in Kowarski's files. IBM, too, stretched its feelers towards CERN,

inviting Kowarski to demonstrations of IBM computers scheduled in Geneva and Lausanne for the end of October (LK1.14). Kowarski went to the Lausanne demonstration and IBM later wrote him that the machine which best fit CERN needs was probably the IBM 701 (LK1.15). Finally, on October 24 the SEA director sent to Kowarski a long description of and offer for the planned CAB 3040, which displayed all the features CERN desired (LK1.16). After prompting from SEA, CERN's director's office replied that they would consider the offer (LK1.17, 1.18). As noted above, the CAB 3040 was never built, so one may wonder whether it was planned primarily with a view of making a deal with CERN.

8. Kowarski's Final Report: the Ferranti Mercury as the Ideal Scientific Computer

After having collected information from fellow scientists and computer manufacturers, Kowarski wrote a second report on computing recommending the purchase of a Ferranti Mercury (CERN/DG/45).¹⁷ The report, dated November 1, was discussed and approved by CERN Directors on November 11, pending approval by the Finance Committee (FC). Before addressing the FC Kowarski decided to ask for the opinion of the Scientific Policy Committee (SPC). He probably assumed that the arguments he put forward would be best appreciated by scientists, and that the FC would not be able to ignore their endorsement. The SPC meeting took place on November 21 and the list of participants was a roll-call of the most prominent particle physicists in Europe: Werner Heisenberg (chairman), Gilberto Bernardini, Patrick Blackett, Niels Bohr, Louis Leprince-Ringuet, and Paul Scherrer (members), Herbert Skinner and Edoardo Amaldi (guests), and Director-General Bakker (CERN/SPC/14). Kowarski and Adams were admitted to the meeting for the discussion on the computer.

The report was 10 pages long, including as an attachment a two-page extract from Anderson's letter. The gist was a recommendation to order a Ferranti Mercury, confirming and strengthening the suggestions from the August meetings. In this section, I will closely analyze the way in which Kowarski summarized the information collected and so set the stage for a decision in favor of the Ferranti Mercury. I do this because I regard the report as Kowarski's careful distillation of current views about computing and computers among those nuclear and particle physicists who had had some experience of them. As already suggested above, two features stand out as characteristic of the positive view of computing in that community: first, a rather clear-cut

17 The report CERN/DG/45 is available online as an attachment to the documents sent by Kowarski to the SPC on November 11, 1955 (CERN/SPC/13). Drafts of the report CERN/DG/45 are found in the CERN Archive (LK1.19). As for all official CERN documents, it was circulated in both English and French versions, and I quote from the English one.

distinction was assumed between business and scientific computers, with the latter viewed as more powerful than the former, even though—or perhaps because—they might not yet be available on the market. Second, scientific computing in nuclear and particle physics was seen as a collective enterprise, requiring for its success a constant sharing of know-how, innovations, and resources. At that specific moment, the Ferranti Mercury happened to be the choice that better fit this emerging image of scientific computing in an emerging field of big science.

In line with physicists' traditional wish for "objective" information, the report started by stating the difficulties of obtaining it in this case: knowledge about computing, wrote Kowarski, was in itself "mathematical and logical," but the information available was biased by commercial interest. To try and gain an unbiased view, besides representatives of European and American firms, trusted peers had also been contacted: CERN staff possessing relevant expertise, Anderson, and "some present or future users (Dr. Howlett of Harwell; Dr. Gradwell of Ferranti's; Dr. Horowitz¹⁸ and Mr. Amouyal of Saclay)" (CERN/DG/45, 2). Further information would be gathered at the conference to be held in Cambridge in December. No mathematicians or computing engineers were consulted: the use of computers in fundamental physics research was a topic in which authority should rest with physicists.

The CERN computer should be a purely binary machine, as opposed to one using decimal or mixed storage formats. The motivation for this choice is interesting: "The binary system was at first adopted for purely electronic reasons, but it is increasingly recognized as the closest approximation to the basic processes of thought and logic" (CERN/DG/45, 3). Thanks to the analogy to the human brain, the computer appeared as more than just another electronic instrument whose details could be left to the technicians. Further desired features were: a large range of memory digits, speed, automatic floating point conversion and, finally, ease of operation. The latter characteristic was important because the physicists who would use the machine were not (yet) skilled programmers. The alternative was to hire experts (i.e., mathematicians) to perform the coding: not only an expensive move, but one which would prevent making computing a part of physics research as a "new experimental technique." Finally, requirements were mentioned that were not purely technical: CERN should purchase a computer similar or identical to those at laboratories engaged in (high-energy) nuclear research. The report states:

As a relatively small organization which cannot hire staff in order to make a contribution to the rapid development of the scientific electronic

18 Jules Horowitz was a prominent nuclear physicist from Saclay. One of Kowarski's handwritten notes with remarks on computers is marked "J.H.," possibly recording a conversation with Horowitz (LK1.20).

computation, CERN cannot afford to play the role of an initial nucleus in some new departure. Hence the necessity of choosing a type which should be identical, or at least extremely close in its programming characteristics, to a machine used by at least one bigger organization whose work lies in a field similar to CERN's.

The extent to which parallel users of the same, or nearly the same, machine can help each other is very much greater in the case of electronic computers than in that of practically any other item of machinery. The field of mutual help goes far beyond a mere exchange of maintenance experience or spare parts; the situation is rather comparable to the adoption of a common language and the possibility of sharing a common, and highly esoteric, library. (CERN/DG/45, 4)

Harwell and Saclay are here not mentioned, but they of course were known to all present. For scientific users such self-help communities were essential, and in this case it had to be a disciplinary-specific community.¹⁹ Kowarski's presentation made clear that CERN was not simply starting to use computers: it was embarking on the shaping and sharing of a new research tool for high-energy (nuclear) physics. Yet CERN at this time was in no position to hire staff to develop computational tools: only some years later, when the laboratory had grown beyond expectations, would it also become a center of scientific computing. The following section of the report commented on the offerings on the market, noting first that US machines, although technically very advanced, were too expensive. Turning to British and French manufacturers, Kowarski took up again the idea that one should try and obtain an objective estimate of which machine fit CERN needs independently of commercial advertising. The procedure he described in the report sounded very much like an application of experimental methodology:

Among the British manufacturers, Ferranti Ltd seems to have put on the market the most up-to-date scientific machines (none completed at the time of writing). Owing to the existence of contacts between their specialists and the prominent Harwell expert (Dr. Howlett), we have asked this firm to help us in clarifying our ideas on the specification (see above). A comparison between two of their currently proposed types, Pegasus and Mercury (also known as Mark II), was taken as a basis of discussion, thus eliminating the complication of a commercial bias. (CERN/DG/45, 5)

Did Kowarski really believe that asking Ferranti to compare two of its own machines would eliminate a commercial bias? I personally see this statement as a way to steer the members of the SPC towards the Mercury. Whatever the case, though, the argumentation once again implied that computers on

19 Such self-help groups started emerging in the 1950s, the first being the SHARE group of IBM users (Ceruzzi 2003, 88).

the market could be classified into two types, and the Pegasus represented the smaller, slower, if cheaper, machines which did not possess most of the features CERN desired. The Mercury, on the other hand, was ideal:

Mercury and ERA²⁰ are the only machines which offer all [required characteristics]. In the case of Mercury, this coincidence has been determined by the continuous interchange of ideas which took place between the scientific users of computing machinery and the Ferranti staff during the last few years, prior to the freezing of the Mercury design. (CERN/DG/45, 5)

The Ferranti Mercury was presented as the best choice because its design had been shaped by physicists, and so provided the best fit for the ideal of a purely scientific machine. From this perspective, the fact that the Mercury was a highly experimental machine made it more, and not less, attractive as a scientific computer. After this comparison, a few words were spent on other British and French products: the Elliott 404, an update of the 401, the EE Deuce and Bull's Gamma were quickly cast in the role of smaller machines, although they were also being used at research institutions. The only other computer that fit CERN's requirements was the CAB 4030 by SEA, which was in an even earlier stage of planning than the Mercury.

In the end, Kowarski pleaded for a budget of up to CHF1 million, which would allow the purchase of a Ferranti Mercury or CAB 3040. Which of the two though? The arguments brought forward in favor of the Mercury were: (1) other scientific institutions had already ordered a Mercury, so CERN would be neither the first nor the only user of the machine; (2) Ferranti had more experience than SEA, thus making a reliable delivery time more probable; and (3) Harwell and Saclay had both ordered a Mercury. The last point carried the most weight: as noted above, CERN was entering not into computing, but into scientific computing and should do so as part of a (budding) community sharing the same tools and profiting from each other's experience. The final section of the report addressed the question of hiring and training staff for the computer, and plans for sharing training programs with Saclay and possibly also Harwell were mentioned.

20 The ERA (Engineering Research Associates) was a former US start-up in computer manufacturing which had been acquired by Remington Rand (Campbell-Kelly et al. 2023, 109-10). This is the only time ERA is mentioned in Kowarski's papers.

9. Reactions to Kowarski's Report and Final Decision

How did the SPC react to the proposal? The draft minutes of the meeting held on November 21 state:

Professor Bernardini stressed the usefulness of an electronic computer in particular for high-energy physics where the complicated nature of the processes necessitates such an instrument. It was thus decided to appoint a sub-committee, consisting of Prof. Amaldi, Prof. Bernardini, Dr. Kowarski and Mr. Adams to consider the proposal and express their opinion on it. (CERN/SPC/14)

The decision was delegated to an appointed subcommittee formed by particle physicists who had experience with computing. The subcommittee met on the next day and confirmed Kowarski's recommendation (LK1.21). The report now went to the Finance Committee (FC) and, since by now the deadline for Ferranti's option was near, an extraordinary ad hoc meeting was scheduled for December 20 (LK1.22). The committee represented CERN member states and, according to the minutes of the meeting, only the British representative H. L. Verry voiced doubts, asking "if it would not be desirable to consider more closely the merits of certain other types of electronic computer, such as the Deuce machine, produced by the English Electric Company, whose cost was much lower" (CERN/FC/108, 3). Verry's doubts seem quite well-founded, but they were met with well-prepared opposition, sketching once again the desired features of scientific computing at CERN:

Dr. KOWARSKI said that he had had an opportunity of studying the distinctive features of the Deuce computer, and had found that these did not meet the technical requirements of CERN. While it was true that this machine would cost about half the price of the Ferranti computer, it would not perform the same work. ... Mr. ADAMS pointed out that a machine of the Deuce type required far more time to perform its work than a Ferranti Mercury computer; and that such loss of time would involve heavier costs in the long run. Moreover, seeing that every type of electronic computer operated in a different "language," the choice of a Deuce machine would make all exchange of information between CERN, Saclay and Harwell more difficult, if not impossible. Obviously, it was most desirable to ensure unhampered exchange of information between these three laboratories engaged in similar work. (CERN/FC/108, 4)

Verry gave up his opposition, and the next day Kowarski wrote to Ferranti confirming that the purchase had been finally approved (LK1.23). He also asked Ferranti for cooperation in searching for candidates for the posts of

“mathematical leader” and relevant staff for the computing services. One day later, he contacted Wilkes in Cambridge, too, on this issue (LK1.24).

10. Epilogue: CERN as a “Shop Window” for European Scientific Computing

When deciding on the purchase of a computer, Kowarski and other CERN scientists had taken a delay in the delivery more or less for granted. When the estimated time of delivery came, though, CERN physicists had developed a series of tasks for the computer, and now found themselves in need of alternative solutions. The same had happened at Saclay, where an IBM 650 had to be purchased while waiting for the Mercury (Amouyal 1990, 220). In March 1957 Kowarski wrote to Bernard Swann, head of Ferranti’s Sales Department, asking for assistance in finding alternative computing facilities until the Mercury came (LK1.25, 1.26). Thus, by early 1957 a need for computational power had developed which had not been present in 1955. The (planned) availability of a computer had created demand, and not the reverse. Swann offered to let CERN use for a reduced fee a Pegasus installed at the Ferranti Computer Centre in London (LK1.27). Eventually, the Mercury was installed only in October 1958 and in the meantime CERN computer work had been sent around not only to the Pegasus in London, but also to an IBM 704 in Paris, Mercuries in Harwell and Manchester, a Deuce in Teddington, and an IBM 650 in Darmstadt (Beck 1964, Borrelli 2019). By 1960 computers were essential instruments for CERN physicists. Soon it was necessary to acquire new machines and Adams, at the time acting CERN Director General,²¹ in December 1960 corresponded with Ferranti on that subject, explaining that it would be difficult to convince CERN physicists to purchase new Ferranti machines, as they still remembered “bad episodes” with the Mercury (DGR14.1). Adams gave no details on the negative experiences, but explained that CERN had recently rented an IBM 709 “which was delivered exactly on time and which was working efficiently after two weeks.” However, as a European laboratory, CERN was set on buying European products and would be looking at Ferranti and EE computers “with a view of selecting the best” (DGR14.1). He underscored how CERN was now central in European computing: “Ferranti should concentrate particularly on CERN computing if they wanted to sell their machine in Europe, since CERN is a shop window for such sales” (DGR14.1).

Thus, CERN physicists had now come to appreciate reliability, though they still aimed at the most powerful machines available. Scientific computing had become a valuable market niche, which, however, Ferranti was not destined to fill: in 1961 CERN acquired an IBM 709, soon replaced it with its transistorized

21 The Director General Bakker had died in April 1960 in a plane crash, and Adams had been appointed to substitute him until the next Director General would be elected.

version IBM 7090, and in 1965 moved on to a CDC 6600. By that time, the old Mercury was being increasingly used for fast data analysis at the PS (Beck 1964).

11. Conclusions

The reconstruction and contextualization of CERN's purchase of its first computer supports the thesis that the diffusion of computing in the sciences in the late 1950s and early 1960s went hand in hand with a process of diversification. More specifically, we have seen how CERN's purchase was not part of a homogeneous trend in the sciences, but was largely shaped by the historical actors and disciplinary communities involved. In a similar way, the period saw the emergence of a varied landscape of cultures of scientific computing comprising not only machines and codes, but also programmers, users and networks for shaping and sharing resources and know-how according to common disciplinary methods and aims.

The results of the case study can be summarized in five more or less tentative conclusions. First of all, we may confidently state that CERN scientists did not endorse the acquisition of a computer because they were facing some problem that only a computer could solve. Indeed, before Kowarski brought up the possibility, it seems no one at CERN had seriously considered acquiring one of the new machines, although at the time they were available on the market and used by various research institutes. As we saw above, in 1953 Adams and Hine had used the ACE computer at the NPL in London to perform computations for the design of the CERN PS, but this fact does not seem to have motivated CERN to consider buying its own computer. In 1954 Adams became director of the CERN PS Division, but there are no records of any interest on his part for computing at this stage, although he was later among the first to support Kowarski's proposal. While this episode might appear strange from today's point of view, it fits the situation in the early 1950s, when even computer experts doubted that there would ever be enough computational tasks in science to occupy more than a few computers. In our case, it might have seemed to CERN scientists that, once the calculations for the design of the PS had been completed, a computer would be of little use.

Here it is also important to stress once more that Monte Carlo event generators for high-energy physics were still in the future and computer-aided data analysis in particle physics was a very new method used in practice only at Chicago, and not necessarily with great success (Borrelli 2019). Therefore, it is not surprising if CERN physicists did not regard a computer as a necessary part of CERN infrastructure. Yet once the idea of buying one was on the table, they quickly took an interest in it, especially those who had already had some experience with computing in the USA or UK. By early 1957, many uses had

been found for the Ferranti computer, which still had to be delivered, and temporary solutions had to be organized to bridge the gap. Here we have an interesting example of the dynamics between tradition and innovation, which gives a new perspective to Jon Agar's reflections on whether computers made tasks possible that had until then been regarded as impossible (Agar 2006). In this case, a computer was purchased with no new specific tasks in mind, but its (potential) availability led to the definition of new functions that only a computer could fulfill, such as the first application of Monte Carlo computations to estimate cross-sections of particle processes (Borrelli 2019).

In the second place, we have seen how computer manufacturers had little or no influence on the choice of machine. Contacts with the French and UK governments helped offers by SEA, Bull and EE be taken into consideration by CERN officials, but the decision to buy a Ferranti Mercury was primarily shaped by the positive opinion of colleagues from nuclear and particle physics. Why was the Mercury specifically chosen, though? The arguments brought forward in its favor were its indeed rather unique advanced features. However, no concrete reason was presented as to why these specific features were needed, and, when the delivery of the Mercury was delayed, computers were used in its place that did not possess them, like the IBM 650, the Pegasus, and the Deuce. From this background, I suggest a third hypothesis, namely that the drive towards more powerful, experimental machines was also linked to the self-image of science, or at least of big science. In his letter, Anderson explained very clearly that the most powerful machines were needed for computations in physics, or at least in particle physics, because physicists would be motivated to engage with computers only if they perceived them as the utmost powerful scientific tool. Anderson underscored how the idea of being able to compute faster than researchers of previous generations motivated students to use computers. Although this view can be understood purely as a wish to optimize resources, the sole focus on computational speed does not necessarily imply optimization of research tasks. Here it may be noted that, according to Agar, speed was the most often mentioned feature which let the computerization of tasks appear desirable (Agar 2006, 881, 885, 887).

My third remark is that the wish to acquire the most powerful, fastest computers can also be seen in analogy to particle physicists' striving towards constantly larger, more powerful accelerators capable of bringing particles to higher and higher speed, and therefore energy. In big science, competition also expressed itself in terms of technological power, and the computer was at least partly being imagined in analogy to the main research tool of the discipline: the accelerator. I have also suggested that, from this point of view, the fact that powerful computers at the time were very expensive, experimental machines whose performance and delivery date were still uncertain made them appear all the more suited to scientific aims, because it set them

apart from more affordable, reliable machines often preferred by business and administration users. In the early 1950s no clear-cut distinction between scientific and business machines yet existed, but physicists drew it to label some computers as machines for business and others as scientific tools whose characteristics were shaped by the scientific community. Physicists' preference for powerful machines also fit well the interests of engineers like those working in the Ferranti laboratories in Manchester, developing a cutting-edge machine against the wishes of their sales department. Nuclear and particle physics laboratories were essential to the commercial success of the Ferranti Mercury, but its follow-ups, the Atlas and Orion, did not fare as well, partly also because Ferranti management had lost interest in its computing branch, which they sold in the early 1960s (Tweedale 1990). At that point, though, other firms had positioned themselves to take advantage of the new market niche of scientific computing: IBM with its 700 series and CDC, a small US firm initially struggling to compete with IBM and UNIVAC, but which would eventually become a major player in that field (Lewis 2017, 30).

My fourth tentative conclusion is that, when computers started being employed in nuclear and particle physics, physicists laid great value in choosing and shaping the machines with as little input from computer experts as possible, relying primarily on the members of their own community for counsel and mutual help. In fact, I have argued that the emergence of a computing community in (high-energy) nuclear physics was a premise, and not a consequence, of the introduction of computers at CERN. This is shown by the fact that an important argument for purchasing a Mercury was that nuclear physics laboratories in Harwell and Saclay had also ordered one: right from the beginning computing at CERN was seen as a collective enterprise going beyond a single laboratory. At that specific moment the Mercury was the machine of choice of the nuclear physics community, and therefore also of CERN. In other words, if the nuclear physics community had not started developing an interest in computing, CERN might have waited longer before entering the field. However, research in high-energy physics soon diversified from that in (low-energy) nuclear physics, and at that time a specific range of computing practices for it emerged, linking CERN more to US laboratories.

Finally, a fifth conclusion emerging from the study is that, in the early 1950s, in the community of European nuclear and particle physicists the view that computers could become valuable scientific instruments for their disciplines was promoted by contacts with and initiatives by Maurice Wilkes and his group in Cambridge. The importance of Wilkes, his collaborators and the Cambridge schools on computing for the development of theoretical and applied informatics has been discussed by the literature (Ahmed 2013), and Mark Priestley's chapter in this volume highlights the influence of Wilkes' group on the Whirlwind researchers at MIT. However, their role in motivating and facilitating

the emergence of cultures of scientific computing largely awaits systematic investigation. The example of nuclear and particle physics suggests that the Cambridge workshops on scientific computing, although they did not—and could not—address the specific needs of those disciplines, nonetheless were of great importance in conveying the message that, if given enough attention, electronic digital computers could be made into cutting-edge, customized scientific instruments.

For their very helpful comments on previous versions of this paper I thank all contributors to this volume, in particular Troy Astarte, Helena Durnová, Jacqueline Léon and Mark Priestley, and I am deeply grateful to Liesbeth De Mol for motivating and inspiring this work. This paper would not have been possible without the advice and assistance of Anita Hollier at the CERN Archive. For financial support and a most stimulating research environment I am greatly indebted to the directors, staff and fellows of the DFG-funded Institute for Advanced Study on Media Cultures of Computer Simulation (MECS), Leuphana University Lüneburg (DFG grant KFOR 1927) and of the Käte-Hamburger-Kolleg "Cultures of Research," RWTH Aachen.

References

Archival Sources

From: CERN Archive, Folder CERN-ARCH-LK-001 (papers of Lew Kowarski)

- LK98.1:* Y[ves] G[oldschmidt]-C[laremont] "Preliminary technical notes on some activities of the experimental group," CERN/LZ, October 15, 1953
- LK98.2:* Letter Raymond (SEA) to Kowarski, 4/6/54
- LK98.3:* Letter Wilson (Ferranti) to Kowarski, 4/6/54
- LK98.4:* Letter Wilson (Ferranti) to Kowarski, 22/6/54
- LK98.5:* Letter Kowarski to Raymond (SEA), 29/6/54
- LK98.6:* Letter Kowarski to Wilson (Ferranti), 30/6/54

From: CERN Archive, Folder CERN-ARCH-LK-098 (papers of Lew Kowarski)

- LK1.1:* Kowarski, Draft note of report to CERN Director General Nr. CERN/DG/27, 1, undated, written between August 2 and 8, 1955. The final version of the report CERN/DG/27 was discussed in a Directors' meeting on August 9 (CERN/SPC/13)
- LK1.2:* Letter Johnson (Ferranti) to Kowarski, 11/10/55 – includes program of seminar on "Application of modern methods of computation in physical research" organized at the Art School in Cambridge in cooperation with Wilkes' Mathematical Laboratory (December 12–13)
- LK1.3:* Letter Stiefel to Kowarski, 16/11/55
- LK1.4:* Copy of letter Anderson to Goldschmidt-Clermont, 11/8/55
- LK1.5:* Kowarski, hand-written notes in French on Amouyal's reports from 2/6/55, undated but written between August and October 1955
- LK1.6:* Letter Johnson (Ferranti) to Kowarski, 11/11/55
- LK1.7:* Letter Chedzoy (EE) to Kowarski, 26/8/55
- LK1.8:* Letter Kowarski to Chedzoy (EE), 5/9/55
- LK1.9:* Letter Chedzoy (EE) to Adams, 1/9/55
- LK1.10:* Letter Chedzoy (EE) to Adams, 13/10/55
- LK1.11:* Letter Kowarski to Pitman (EE), 18/10/55
- LK1.12:* Letter Pitman (EE) to Kowarski, 31/10/55
- LK1.13:* Letter Wurmser (Bull) to Kowarski, 26/9/55, in attachment a document from Bull with specifications of the Gamma Tambour (dated 21/9/55)
- LK1.14:* Letter IBM Suisse (name unreadable) to Direction of CERN, 21/10/55
- LK1.15:* Letter IBM Suisse (name unreadable) to Kowarski, 4/11/55

- LK1.16: Letter Raymond (SEA) to Kowarski, 24/10/55
 LK1.17: Letter Raymond (SEA) to CERN Director, 14/11/55
 LK1.18: Letter Bertrand (Assistant Director General CERN) to Raymond (SEA), 18/11/55
 LK1.19: Kowarski, notes for and drafts of a text which will be finalized as the report to the Director General Nr. CERN/DG/45, dated 11/11/55 and available online
 LK1.20: Kowarski, hand-written note in French, probably of a conversation on computing. It contains remarks similar to LK1.5 and is marked J.H. (Jules Horowitz?). The notepaper is from CNR (Centro Nazionale delle Ricerche) Roma, i.e., the National Centre of Research in Rome.
 LK1.21: Shortened version of CERN/DG/45, report of ad hoc subcommittee 22/11/55, extracts from Ferranti and SEA quotations, marked as annexes I to IV, very probably attached to the report circulated by Kowarski among the members of the Financial Committee on 8/12/55 in preparation for the committee meeting
 LK1.22: Letter Richemond to members of FC, 8/12/55
 LK1.23: Letter Kowarski to Johnson (Ferranti), 21/12/55
 LK1.24: Letter Kowarski to Wilkes, 22/12/55
 LK1.25: Letter Kowarski to Swann (Ferranti), 1/3/57
 LK1.26: Letter Kowarski to Swann (Ferranti), 12/3/57
 LK1.27: Letter Swann (Ferranti) to Kowarski, 14/3/57

From: CERN Archive, Folder CERN-DG-R-1-4 (Director General)

DGR14.1: Letter Adams to Hall (Ferranti), 21/12/60

Online Archival Sources

- CERN/DG/45: Electronic Computer (Second Report to the Director-General) [by L. Kowarski], 1/11/55, attached to CERN/SPC/13 *CERN Document Server*. CM-P00098652-f, November 11, 1955. Accessed February 2025. <https://cds.cern.ch/record/1124358>.
 CERN/SPC/13: Lew Kowarski "Proposal to Purchase an Electronic Computer," 11/11/55, circulated in preparation of the 3rd Meeting of the Scientific Policy Committee scheduled for 21/11/55. *CERN Document Server*. CM-P00098652-f, November 11, 1955. Accessed August 2024. <https://cds.cern.ch/record/1124358>.
 CERN/SPC/14: Draft Minutes of the 3rd Meeting of the Scientific Policy Committee, 21/11/55. *CERN Document Server*. CM-P00094785-e, December 7, 1955. Accessed February 2025. <https://cds.cern.ch/record/1124360>.
 CERN/FC/108: Draft Minutes of the 7th Meeting of the Finance Committee, 20/12/55, *CERN Document Server*. CM-P00079586-e, 1955. Accessed February 2025. <https://cds.cern.ch/record/20297>.

References

- Agar, Jon. 2006. "What Difference Did Computers Make?" *Social Studies of Science* 36: 869–907. <https://doi.org/10.1177/0306312706073450>.
 Ahmed, Haroon. 2013. *Cambridge Computing: The First 75 Years*. London: Third Millennium Publishing.
 Amouyal, Albert. 1990. "The Beginnings of Computing Activities at the Atomic Energy Authority, 1952–1972." *Annals of the History of Computing* 12: 219–25. <https://doi.org/10.1109/MAHC.1990.10045>.
 Anderl, Sibylle. 2019. "Implicit Changes of Model Uses in Astrophysics, Illustrated on the Paris-Durham Shock Model." *NTM* 27: 515–46. <https://doi.org/10.1007/s00048-019-00225-8>.
 Anderson, Herbert. 1986. "Scientific Uses of the MANIAC." *Journal of Statistical Physics* 43: 731–48. <https://doi.org/10.1007/BF02628301>.
 Anderson, Herbert, William C. Davidon, Maurice Glicksman, and Ulrich E. Kruse. 1955. "Scattering of Positive Pions by Hydrogen at 189 Mev." *Physical Review* 100: 279–87. <https://doi.org/10.1103/PhysRev.100.279>.

- Aspray, William, and Christopher Loughnane. 2019. "Foregrounding the Background: Business, Economics, Labor, and Government Policy as Shaping Forces in Early Digital Computing History." In *Exploring the Early Digital*, edited by Thomas Haigh, 159–78. Cham: Springer.
- Battimelli, Giovanni, and Giovanni Ciccotti. 2018. "Berni Alder and the Pioneering Times of Molecular Simulation." *The European Physical Journal H* 43: 303–35. <https://doi.org/10.1140/epjh/e2018-90027-5>.
- Battimelli, Giovanni, Giovanni Ciccotti, and Pietro Greco. 2020. *Computer Meets Theoretical Physics: The New Frontier of Molecular Simulation*. Cham: Springer. <https://doi.org/10.1007/978-3-030-39399-1>.
- Beck, Frank. 1964. "Computers at CERN." *CERN Courier* 4 (June): 73–77.
- Biener, Klaus. 1999. "Alwin Walther – Pionier der praktischen Mathematik." *RZ-Mitteilungen* 18: 60–62. <https://doi.org/10.18452/6275>.
- Birkhoff, Garrett. 1990. "Fluid Dynamics, Reactor Computations, and Surface Representation." In *A History of Scientific Computing*, edited by Stephen Nash, 63–87. Reading, MA: Addison–Wesley.
- Bonfanti, Corrado. 2012. "Information Technology in Italy: The Origins and the Early Years (1954–1965)." In *Reflections on the History of Computing: Preserving Memories and Sharing Stories*, edited by Arthur Tatnall, 320–47. Berlin: Springer.
- Borrelli, Arianna. 2019. "Program FAKE: Monte Carlo Event Generators as Tools of Theory in Early High Energy Physics." *NTM* 27: 479–514. <https://doi.org/10.1007/s00048-019-00223-w>.
- . 2022. "Der Aufstieg von arXiv: Netzwerkeffekte und wissenschaftliche Kommunikation." In *Zur Geschichte des digitalen Zeitalters*, edited by Ricky Wichum and Daniela Zetti, 165–83. Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-34506-8_9.
- Borrelli, Arianna, and Janina Wellmann. 2019. "Computer Simulations Then and Now: An Introduction and Historical Reassessment." *NTM* 27: 407–17. <https://doi.org/10.1007/s00048-019-00227-6>.
- Brown, Laurie, and Lillian Hoddeson, eds. 1986. *The Birth of Particle Physics*. Cambridge: Cambridge University Press.
- Brown, Laurie, Max Dresden, and Lillian Hoddeson, eds. 1989. *Pions to Quarks: Particle Physics in the 1950s*. Cambridge: Cambridge University Press.
- Campbell-Kelly, Martin. 2019. "The Evolution of Digital Computing Practice on the Cambridge University EDSAC, 1949–1951." In *Exploring the Early Digital*, edited by Thomas Haigh, 117–34. Cham: Springer.
- Campbell-Kelly, Martin, William Aspray, Jeffrey R. Yost, Honghong Tinn, Gerardo Con Díaz, and Nathan Ensmenger. 2023. *Computer: A History of the Information Machine*. 4th edition. Boulder, CO: Perseus Books.
- CERN Courier. 1962. "Gilberto Bernardini." *CERN Courier* 2 (July): 7.
- Cerulus, Frans. 2012. "Remembering Wim Klein." *CERN News* (blog). December 5, 2012. <https://home.web.cern.ch/news/news/cern/remembering-wim-klein>.
- Ceruzzi, Paul E. 2003. *A History of Modern Computing*. Cambridge, MA: MIT Press.
- Cortada, James W. 2012. *The Digital Flood: Diffusion of Information Technology Across the United States, Europe, and Asia*. Oxford: Oxford University Press.
- de Chadarevian, Soraya. 2002. *Designs for Life: Molecular Biology After World War II*. Cambridge: Cambridge University Press.
- Eyßell, Manfred. 2010. "Heinz Billing – Der Erbauer der ersten deutschen Elektronenrechner." *GWDG-Nachrichten* 4: 9–36.
- Galison, Peter. 1997. *Image and Logic: A Material Culture of Microphysics*. Chicago: University of Chicago Press.
- Gandy, Anthony. 2013. *The Early Computer Industry: Limitations of Scale and Scope*. Berlin: Springer.
- Gugerli, David, and Ricky Wichum. 2021. *An den Grenzen der Berechenbarkeit: Supercomputing in Stuttgart*. Zürich: Chronos.
- Haigh, Thomas, ed. 2019. *Exploring the Early Digital*. Cham: Springer. <https://doi.org/10.1007/978-3-030-02152-8>.

- Haigh, Thomas, and Paul E. Ceruzzi. 2021. *A New History of Modern Computing*. Cambridge, MA: MIT Press.
- Heisenberg, Werner. 2001 [1958]. "Letter to Wolfgang Pauli. February 27th 1958." CERN Document Server. February 20, 2001. <https://cds.cern.ch/record/85522>.
- Hermann, Armin. 1987. "Appendix 1: Who's Who in the Foundation of CERN." In *History of CERN Vol. 1: Launching the European Organization for Nuclear Research*, edited by Armin Hermann, Lanfranco Belloni, and John Krige, 545–66. Amsterdam: North-Holland.
- Hermann, Armin, Lanfranco Belloni, John Krige, eds. 1987–1990. *History of CERN*. 2 vols. Amsterdam: North-Holland.
- Hocquet, Alexandre, and Frédéric Wieber. 2017. "Only the Initiates Will Have the Secrets Revealed: Computational Chemists and the Openness of Scientific Software." *IEEE Annals of the History of Computing* 39: 40–58. <https://doi.org/10.1353/ahc.2017.0030>.
- Hughes, Jeff. 2002. "Radioactivity and Nuclear Physics." In *The Cambridge History of Science: Volume 5: The Modern Physical and Mathematical Sciences*, edited by Mary Jo Nye, 350–74. Cambridge: Cambridge University Press.
- Howlett, Jack. 1979. "Computing at Harwell." In *25 Years of Theoretical Physics at Harwell: 1954–1979*. <http://www.chilton-computing.org.uk/acl/literature/reports/p009.htm>.
- Kowarski, Lew. 1971. Interview of Lew Kowarski by Charles Weiner on May 15, 1971, Niels Bohr Library & Archives, American Institute of Physics, College Park, MD, USA, www.aip.org/history-programs/niels-bohr-library/oral-histories/4717-7.
- Krige, John. 1987a. "Survey of Developments." In *History of CERN Vol. 1: Launching the European Organization for Nuclear Research*, edited by Armin Hermann, Lanfranco Belloni, and John Krige, 209–36. Amsterdam: North-Holland.
- . 1987b. "Case Studies of Some Important Decisions." In *History of CERN Vol. 1: Launching the European Organization for Nuclear Research*, edited by Armin Hermann, Lanfranco Belloni, and John Krige, 237–92. Amsterdam: North-Holland.
- Lavington, Simon. 2011. *Moving Targets*. London: Springer.
- Lawson, John D. 1997. "Early Synchrotrons in Britain, and Early Work for CERN." In *50 Years of Synchrotrons*, edited by J. D. Lawson and Giorgio Brianti, 1–28. Geneva: CERN.
- Lewis, Nicholas. 2017. "Purchasing Power: Rivalry, Dissent, and Computing Strategy in Supercomputer Selection at Los Alamos." *IEEE Annals of the History of Computing* 39: 25–40. <https://doi.org/10.1109/MAHC.2017.3481344>.
- MacKenzie, Donald. 1991. "The Influence of the Los Alamos and Livermore National Laboratories on the Development of Supercomputing." *Annals of the History of Computing* 13: 179–201. <https://doi.org/10.1109/MAHC.1991.10014>.
- Mori, Elisabetta. 2019. "Coping With the 'American Giants': Mergers and Attempted Partnerships in the European Computer Industry (1962–1964)." *IEEE Annals of the History of Computing* 41: 83–96.
- . 2022. "Pursuing Efficiency in Business: The Technological and Commercial Development of LEO Computers." PhD Thesis, Middlesex University London, Department of Computer Science.
- Mounier-Kuhn, Pierre. 1989. "Bull: A World-Wide Company Born in Europe." *IEEE Annals of the History of Computing* 11: 279–97. <https://doi.org/10.1109/MAHC.1989.10045>.
- . 1998. "Calculateurs électroniques et nouveaux systèmes d'armes: Interactions Armées/Recherche/Industrie (1946–1959)." In *La IVe République et les problèmes d'armement*, edited by Maurice Vaïsse, 376–405. Paris: Centre d'études d'histoire de la défense.
- . 2003. "Bull, la recherche publique et les marchés internationaux. Développement de calculateurs et coopération franco-italienne (1949–1980)." *Mélanges de l'école française de Rome* 115: 715–44. <https://doi.org/10.3406/mefr.2003.10061>.
- . 2010. *L'informatique en France: de la seconde guerre mondiale au Plan Calcul: L'émergence d'une science*. Paris: Presses de l'université Paris-Sorbonne.
- Nash, Stephen, ed. 1990. *A History of Scientific Computing*. Reading, MA: Addison-Wesley.

- Park, Buhm Soon. 2003. "The 'Hyperbola of Quantum Chemistry': The Changing Practice and Identity of a Scientific Discipline in the Early Years of Electronic Digital Computers, 1945–65." *Annals of Science* 60: 219–47. <https://doi.org/10.1080/00033790.2018.12016717>.
- Pestre, Dominique. 1987. "The Period of Conflict, August–December 1951." In *History of CERN Vol. 1: Launching the European Organization for Nuclear Research*, edited by Armin Hermann, 147–78. Amsterdam: North-Holland.
- Raymond, Francois-Henri. 1989. "An Adventure with a Sad Ending: The SEA." *Annals of the History of Computing* 11: 263–77. <https://doi.org/10.1109/MAHC.1989.10049>.
- Seidel, Robert W. 1998. "'Crunching Numbers' Computers and Physical Research in the AEC Laboratories." *History and Technology* 15: 31–68. <https://doi.org/10.1080/07341519808581940>.
- Speiser, Ambros. 2004. "Konrad Zuses Z4 und die ERMETH: Ein weltweiter Architektur-Vergleich." In *Geschichten der Informatik: Visionen, Paradigmen, Leitmotive*, edited by Hans Dieter Hellige, 171–94. Berlin: Springer.
- Stevens, Hallam. 2013. *Life Out of Sequence: A Data-Driven History of Bioinformatics*. Chicago: The University of Chicago Press.
- Tweeddale, Geoffrey. 1992. "Marketing in the Second Industrial Revolution: A Case Study of the Ferranti Computer Group, 1949–63." *Business History* 34: 96–127. <https://doi.org/10.1080/00076799200000005>.
- Varenne, Franck. 2010. *Formaliser le vivant: Lois, théories, modèles?* Paris: Hermann.
- Wilkes, Maurice, David Wheeler, and Stanley Gill. 1951. *The Preparation of Programs for an Electronic Digital Computer: With Special Reference to the EDSAC and the Use of a Library of Sub-routines*. Cambridge, MA: Addison-Wesley.
- Yood, Charles N. 2013. *Hybrid Zone: Computers and Science at Argonne National Laboratory, 1946–1992*. Boston, MA: Docent Press.

SCIENTIFIC MANAGEMENT

BUSINESS COMPUTING

J. LYONS & CO.

FLOW DIAGRAMS

LEO COMPUTERS

The Influence of Organization and Methods on Early Business Computing

Elisabetta Mori

This paper highlights the influence of Scientific Management and Organization and Methods (O&M) on early business computing by using the case of J. Lyons and Co. and its subsidiary LEO Computers. LEO (Lyons Electronic Office) was a series of main-frame computers developed for business purposes between 1947 and the mid 1960s in the UK. The present analysis elaborates on Lyons' experience in O&M, explaining how these practices also affected computer design and programming practice. LEO was conceived as an adaptation of the EDSAC computer, designed at the University of Cambridge for scientific purposes and operational in 1949. The approach taken to programming the two machines was different: EDSAC used a subroutine library of mathematical calculations, while Lyons wrote more complex master routines to fulfill their business needs. They designed

these using flow diagrams, which they were familiar with because of their experience in O&M.

1. Introduction

The capability to manipulate and manage information has always been of critical importance in business: numerical calculations, organized collection of data, and analysis of information for decision-making are fundamental processes for companies and managers, together with the regulatory requirements of accounting and auditing information for taxation and other purposes. These tasks were performed manually until some form of automation arrived: tools such as mechanical calculators and cash registers first, and punch card equipment later, helped business and were employed in offices from at least the 19th century. The appearance of computers for commercial applications in the 1950s inexorably began to empower and improve business processes. One of the earliest pathways that led to the development of computers for the office, beyond scientific and military purposes, is represented by the case of LEO Computers. These were a series of machines for business applications developed from 1947 by J. Lyons & Co., a well known British food and catering firm, with a history of innovation in management.¹

Some 30 years before the establishment of LEO Computers, Lyons had developed a culture of research and innovation in their manufacturing processes and clerical procedures, seeking the most effective and efficient ways of developing and controlling their business. Lyons followed the principles and methods of Scientific Management and the Organization and Methods movements. Their experience was not limited to clerical procedures, but it was strong in the field of production management. This paper argues that O&M practices in place at Lyons were influential in the computer practice at LEO.

To achieve a working machine in the shortest time possible, in 1947 Lyons opted for a joint venture with Cambridge University, which was developing its EDSAC (Electronic Delay Storage Automatic Calculator) computer. Lyons adapted the Cambridge machine's architecture, designed for scientific computing, to make it suitable for their business purposes. Their quest for efficiency through improvement of processes and methods, combined with their philosophy of vertical integration, led to the construction of their

1 The history of LEO Computers was not popularized in the history of computing until the 1990s, when two books were published—Bird (1994) and Caminer et al. (1996)—written by former Lyons and LEO employees; these were followed by Ferry (2004), a popular science book which contributed to making LEO story known among the general public. This chapter is based on a development of two chapters of the PhD thesis on LEO Computers by the author Mori (2022).

first computer, LEO I, operational in 1951, and to the establishment of LEO Computers Limited in 1954 as a reaction to outside demand for the purchase of LEO machines.

The role of Lyons in the British O&M movement and their use of flowcharts in business administration affected the development of all aspects of the system, including programming techniques and checking routines. Whereas the flowcharting method as a tool to support programming was almost discarded in the EDSAC practice, at LEO instead it represented a common discipline and was extensively used. Flowcharts helped in the understanding and planning of their business procedures and their computer applications.

This paper shows how early business computing was influenced by the work of the pioneers of O&M—Frederick Taylor and Lillian and Frank Gilbreth in particular—as evidenced by Lyons’ deployment of Time and Motion Studies and their use of the flowcharting technique. These methods had already been extensively used at Lyons for administrative processes to avoid wastage and improve efficiency long before the firm entered the computer field.

2. Organization and Methods

Organization and Methods (or O&M study) is the continuous checking of workflows at every stage of an industrial production or office management process. Known in the USA also as Work Simplification, these methods aimed at continuous research for the improvement of management and production at every level.²

On October 12–14, 1911, the pioneers in Scientific Management gathered at their first conference in the field at Amos Tuck School, Dartmouth College, in New Hampshire. They had all just published their seminal works: *The Principles of Scientific Management* (Taylor 1911), *Work, Wages and Profits* (Gantt 1910), *Efficiency as a Basis for Operation and Wages* (Emerson 1909), *Field System* (Gilbreth 1907), and *Bricklaying System* (Gilbreth 1909). The principles of systematic and Scientific Management began to make inroads and entered public discourse in the UK before the First World War. Between the end of the 19th century and the early 1920s a new philosophy of management based

2 Several case studies on the introduction of technological and organizational innovations to improve communication and control in business in the UK and USA have been analyzed by scholars like Agar (2003), Chandler (1962, 1977), and Yates (1989). In particular, Yates’ case studies introduced and adopted new technologies to acquire and record data, in order to base their strategic decisions on the collected information. Jon Agar traced the importance of the Treasury O&M in the computerization of government work in the UK; he shows that O&M played a major role also in the private sector and, specifically, at Lyons (Agar 2003, 293–342). Drawing on Agar, this paper elaborates on Lyons and O&M.

on efficiency arose and spread, and “efficiency” became a catchword in the business world and beyond.³

The avoidance of inefficiency was at the basis of the research of Frederick Winslow Taylor’s (1856–1915) *Scientific Management* (Taylor 1911). His ideas evolved during his lifetime and continued to be influential after his death, developing identities of their own in Europe and Japan (Nelson 1992). Taylor wrote several papers based on his discussions with factory managers. He attempted to transform management from an oral tradition to a set of written techniques based on counting, timing, and measuring. His complex set of methods could be applied to factories but also labor-intensive clerical work. His time studies involved breaking down each job into parts, then timing and rearranging them according to the most efficient combination.

Taylor’s time studies were particularly influential but also received criticism from the unions and from his colleagues, who condemned the lack of actual scientific methodology, arguing that they relied heavily on individual interpretations of what workers actually do. Moreover, Taylor’s attempt to maximize the productivity of a job did not take into account the physiological and psychological cost sustained by the workers themselves. By contrast, Lillian Moller Gilbreth (1878–1972) introduced psychology to management studies and with her husband and colleague Frank Bunker Gilbreth (1868–1924) proposed motion studies, a method based upon the repetitive observation and analysis of work motions. While measuring the time to accomplish a task, the method also included filming the details of the worker activities and their body postures. Filming helped analyze the task of the worker and point out possible improvements. They used film also to train workers about the best way to perform their work (Gilbreth 1911, 1912; Gilbreth and Gilbreth 1917).⁴

In 1921 the Gilbreths co-authored the paper *Process Charts: First Steps in Finding the One Best Way to Do Work*, presented at the Annual Meeting of the American Society of Mechanical Engineers (ASME) (Gilbreth and Gilbreth 1921). The paper introduced their first structured method for documenting process flow. In their research for “one best way,” the Gilbreths developed a collection of tools that would become the foundation of the industrial engineering discipline. The process chart was one of these tools, “a device for visualizing a process as a means of improving it” (Gilbreth and Gilbreth 1921, 3). Its aim was double: representation and improvement. Initially the Gilbreths devised a broader

3 The *Bibliography of Industrial Efficiency and Factory Management*, compiled in 1920, counts approximately 3,500 references, classified under 63 subject headings (Cannons 1920).

4 Frank and Lillian Gilbreth wrote in collaboration, but Lillian’s name was not included until after she earned her PhD in 1915. Frank Gilbreth died of a heart attack in 1924, at age 55. Lillian outlived him by 48 years, continuing the couple’s practice. She gave it a stronger distinctive character, emphasizing the human and psychological element (Price 1992, 70–71).



[Fig. 2] A variation of Flow Process Chart with pre-printed symbols. It shows the present and the proposed method, with a summary of cost savings. (Source: ASME 1947, 16. Public domain)

consultant, working for the Eastman Kodak company among others. Following the Gilbreths' lesson, during the 1930s he experimented using pictures. In 1932 he published his book *Common Sense Applied to Motion and Time Study*, in which he described the concept of Work Simplification, later popularized at his conferences in Lake Placid, New York, beginning in 1937. When offering solutions to improve efficiency in processes he advocated the use of charts as a basis for further improvements, indicating where periodical reviews should be made. In his view, process charts enabled the practitioner to evaluate and reject new solutions unless they were "really better" (Mogensen 1932, 40).

Charles Eugène Bedaux (1886–1944) was a French-born American entrepreneur who became a well-known industrial consultant, with a large clientele on both sides of the Atlantic. He made his fortune developing the "Bedaux System," which implemented the work measurement aspect of Scientific Management. The system was influential in the USA in the 1920s and expanded to Britain in particular during the 1930s and 1940s, improving the workflow both in the food chain and in the administrative field (Kreis 1992).

3. Lyons' Systems Research Office

The discourse on efficiency and Scientific Management developing in the USA became influential in the UK and in Lyons as well. Lyons had very small margins for each meal they served, hence every transaction was recorded and analyzed to check and improve the profits. A small error in cost accounting could have been fatal for profits. Lyons felt that one way to improve the efficiency of their business was to employ managers with an academic and scientific background. Among them there was John Richardson Mainwaring

Simmons (1902–1985), who received first class honours in mathematics at Cambridge. Initially employed as a statistician and management trainee, Simmons remained with Lyons for 45 years, becoming a director and “chief comptroller,”—the company’s manager in charge of the clerical activities.⁵ When he joined Lyons in 1923 the Checking Department took care of the adding up and controlling of millions of waitresses’ bills.

There were some 300 women eventually doing nothing but adding up and checking the adding up of waitresses’ bills. It cried out for some kind of automatic machine to take over this work, which was not fit for human beings to undertake ... Paradoxically by the end of the war, in 1947 ... all the waitresses had disappeared, so the real need for the computer for that kind of job had also disappeared; but it was in our blood.⁶

In 1931 Simmons hired Thomas Raymond Thompson (1907–1976), and established the Systems Research Office, inspired by the work of Taylor, the Gilbreths, Mogensen, and Bedaux.⁷

The principal aim of the Lyons’ Systems Research Office was to analyze, re-engineer, and improve business processes such as invoicing and stock control. Its activities led to innovative developments in Lyons’ clerical procedures: a team of experts was formed, who later would be part of LEO Computers’ core group. Simmons and Thompson centralized the clerical offices spread across many departments, and established three main units: the Checking Department, the Stock Department, and the Accounts Department. The Checking Department, renamed “Catering Office” after the Second World War, controlled the catering records and the teashops’ cash transactions, including every bill. The teashop order stubs were retained too, hence every transaction could be traced and analyzed by its 450 clerks. A prime function was to provide information for senior management and to explain variances between forecasts and actual outcomes. The Stock Department, renamed “Statistics Office” after the Second World War, employed about 400 clerks, and organized the records of raw materials and equipment. Its roles included the provision of management information and explanations of anomalies such as overspending on raw materials. The Accounts Department administrated payrolls, sales, and purchase receipts. This centralized office was crucial to identifying and solving issues that, back in the 1920s, had to be divided into

5 See Bird (2000, 306) and Ferry (1985). A comptroller is responsible for supervising the accounting and financial reporting of a company.

6 John Simmons, interviewed by Christopher Evans, circa 1970s. Copyright 1980, Trustees of the Science Museum, London. Published in Caminer (1996, 364).

7 Not all the scientific management solutions were positively received by Lyons employees: when Lyons introduced a bonus scheme based on the Bedaux system into most of their factories, it was intensely disliked by the staff (Bird 2000, 302).

separate smaller units, divided among bakery, tea, works services, and so on. It employed about 600 clerks (Simmons 1962).

The three departments instituted by Simmons provided a valuable source of information to improve the business, analyzing and understanding the trading patterns. Based on the information collected Simmons could provide weekly or quarterly reports, the White Books, to Lyons' board. Simmons reorganized the equipment of these departments, implementing mechanization to solve the increasingly complex and costly clerical work: by 1934 the Checking Department had about 100 mechanical Burroughs calculators, and 150 bookkeeping machines were used in other Lyons' departments (Bird 2000, 306–307). Simmons and the Systems Research Office explored a variety of office technology: new equipment was introduced only after rigorous testing, and only if the tests proved that they would enhance efficiency in the process. This is the reason why Lyons at some point examined punch card equipment and discarded it on cost-effectiveness grounds (Aris 2000).

As an example of his scientific approach to business management, Simmons recognized that any transfer of data from one document to another was both inefficient and a major source of error. In the study in 1935 he reorganized the Wholesale Bakery Sales invoicing: instead of producing several copies of the client orders, such as invoice, delivery note, packing note, and so on, the customer's order itself was turned into the invoice, adding the price of each item and the total. The order/invoice was sent back to the customer, after taking a picture of it through the use of the Recordak photographic system. This was introduced by the Eastman Kodak Company in 1928 and consisted of a photographic machine and a film projector. At Lyons it was used to record the office copy of the order/invoice on miniature film (16 or 35 mm) (Mills and Standingford 1949, 59–61). In such a way the records occupied very little space and could be referred to by projecting them on a screen projector, providing a durable, extremely compact, and easily accessible file records method.⁸

The Systems Research Office's goal was to identify the appropriate technologies to fulfill Lyons' needs. This was achieved by dividing each job into its functional parts; through the analysis of each of them, the Systems Research Office would identify the right methods and machinery for each sub-task and assign a standard cost to identify the possible cost of each product. The forecast was then compared against the actual data and the analysis shared with the decision-makers.

8 The first practical use of commercial microfilm was designed to make permanent film copies of all bank records. It was invented by George McCarthy, a New York City banker, who was issued a patent for his Checkograph machine in 1925. See also Blaschke (2016).

Simmons' reputation in office O&M progressively grew. In 1933 he joined the British Office Manufacturers Association (OMA).⁹ Lyons participated in the OMA annual conferences: for example, in 1937 Geoffrey J. Mills, the Systems Research Office supervisor at Lyons, gave a talk entitled "Clerical Research" on the O&M procedures in place at Lyons at the OMA annual conference in Bath.¹⁰

We have information about the Lyons' methods via their internal publications, such as the report on *Theory and Practice of Labour Planning*.¹¹ This booklet explained Lyons' company philosophy and organization on the topic. There were two main broad divisions: planning—i.e., studying the problem beforehand; and operating—i.e., getting things done. Managers were usually primarily concerned with operating, hence it was considered of great advantage to appoint specialists dedicated to planning full time. The report stated: "The staff in the Planning Office ... help the operators to accomplish their task in the simplest way possible." Nevertheless, responsibility remained with the manager to accept, modify, or reject the plan. Echoing Mogensen and the Gilbreths, the stress was on optimization of the workflow and simplification. The goal of the Planning Office was the improvement of efficiency. They identified main causes of wastage as: "wrong methods, unbalanced teams, incorrectly designed equipment, bad flow of work, and delays of other kinds."

After the Second World War Geoffrey Mills continued his dissemination of O&M techniques, and he became the author of the manual *Office Organisation & Methods* (Mills and Standingford 1949), published in the UK by Pitman in 1949.¹² The following editions of the book were co-authored with Oliver W. Standingford, another Lyons' senior employee, whose contribution was fundamental in triggering Lyons' decision to enter the computer field in 1947.

9 In 1936 the OMA was renamed the Institute of Office Management. The same year Simmons became its vice-president and then chairman from 1938 to 1950. He was president in 1944–1950, fostering the role of computers as a business tool, holding the title of honorary vice-president until his death in 1985. In 1972 the Institute of Office Management was renamed the Institute of Administrative Management (IAM) (Bryant and Land 2020; Gibson 1991). Lyons' involvement in the Institute was crucial and Simmons' reputation was strategic.

10 Apparently, there was no official printed version of the paper. The Modern Records Centre at the University of Warwick keeps a document concerning the possible presentation of a similar paper at the 7th International Congress for Scientific Management in Washington, DC in 1938. However, this paper, entitled "The Methods specialist" by Mr. G. J. Mills and Mr. R. J. W. Stubbings was "never completed and consequently could not be submitted." This document is dated March 11, 1938 and it is archived at MRC MSS.200/F/2/S2/10/26.

11 J. Lyons & Co., *The Theory and Practice of Labour Planning*, January 1943, revised July 1945, The London Archives, ACC/3527/208.

12 Established in 1886, Sir Isaac Pitman and Sons was to become one of the world's leading educational publishers. Chairman of the publisher from 1934 to 1966 was I. James Pitman (1901–1985), who was director of Treasury O&M between 1943 and 1945 (Agar 2003).

Although the manual doesn't provide any specific example of a flowchart, it is possible to find several similarities with LEO's ways of collecting, recording, and storing of data and programming methodologies, in particular with the practice of independent error checking.

Time and motion studies were extensively used at Lyons to avoid wastage and improve efficiency. Leo Fantl (1924–2000), who joined Lyons in 1948 as a trainee in the Planning Office, remembers that the role of his department was to advise management on new methods and techniques, as well as carrying out detailed activities and observations on time and motion studies. This meant observing each step in a repetitive operation, timing it, and analyzing the level of effort for each step. These observations might involve the passenger traffic in the lifts at the Lyons Cumberland Hotel, ice-cream deliveries, tray washing at Elms House, operations in the Bread Bakery, or a study of office cleaning (Caminer et al. 1998, 157–58).

These practices were inherited also by LEO Computers. Kathleen Fisher (née Keen, b. 1936) who was hired by Lyons aged 16, and whose career evolved from data preparation to programming and training, remembered when Lyons applied the time and motion study techniques to measure punching:

Yes, we had to measure how much we punched and how long it took. And I think what they were doing probably was trying to decide how many people they would actually need to punch certain amounts of data in the future when they had a lot of stuff to do. But we were bright girls and we were very fast so I don't know if they always managed to attain that sort of speed. But yeah, that was good fun as well trying to beat what you did last time.¹³

The Second World War disrupted Lyons' business at many levels. Raw materials were in short supply, demand diminished, shops closed, and many key personnel had to join military operations. Once peace was declared, the post-war economy let activities progressively flourish again, increasing business volume. In Simmons' words:

And then the war was upon us—a long period of disorganisation and distress, and worse. But, as happened elsewhere, it also provided a hot-house in which new ideas could be forced. The double disruption of our normal activities—the disorganisation at the beginning of the war and the reorganisation at its end, encouraged us to think out anew all our theories on organisation. (Simmons 1962, 23)

The post-war boost in business caused a bigger volume of transactions at Lyons. This situation called for a renewed demand for information management. Simmons soon realized that they had to improve the pre-war

13 Kathleen Fisher, private communication to Elisabetta Mori, April 2019.

systems. The time was right for the advent of computers in business data processing. Lyons' pursuit of efficiency, their philosophy of vertical integration, reliance on the principles of Scientific Management, and adoption of O&M procedures were not only key factors for Lyons' decision to enter the computer business, but also affected the practice around the LEO machines.

4. North American Contacts

After the disruptions of the Second World War Lyons' business flourished again and created a renewed demand for improving the management of their business processes, with an emphasis on information. Lyons knew that they had to keep up to date with improvements in office management and food production. To explore war-time and post-war developments in office procedures, catering, food production and distribution they sent two senior employees on a fact-finding trip to the USA and Canada. Oliver Standingford and Thomas R. Thompson traveled from May to July 1947. While they visited major businesses, they also discussed office systems and control with a number of managers at the National Office Management Association (NOMA) in Philadelphia and attended their annual conference in Cincinnati. Standingford and Thompson's survey included the latest developments in computers. It is important to note that in 1947 no commercial computer dedicated to data processing was available on the market, though during the trip they discovered that the giant Prudential Insurance Company had plans, unrealized at that time, for using a computer for its premium billing task. During their overseas trip Thompson and Standingford envisaged the possibility of utilizing computers to help Lyons' clerical procedures and enquired about the potential of electronic machines in the office. On their return they submitted a report on various themes: clerical systems, office administration and office equipment, among others.¹⁴

Section D of the report is dedicated to "Electronic Machines in the Office," including an evaluation of what advantages Lyons might gain from a computer in their business.¹⁵ Lyons had already compared the functions that could be put on a computer to those undertaken by conventional punch card equipment: they had discarded the use of punch card systems on cost-effectiveness grounds. They observed that a digital computer was faster, more flexible in what could be done, and would require less human intervention. Interestingly, Lyons already noticed that input of information to the machine was time sensitive:

14 Thompson and Standingford (1947). Extracts from the report are available in Caminer et al. (1998, 337–59).

15 In the report, computers are referred to as "electronic machines" or "computers."

The machine performs its calculations at speeds which can only be expressed in terms of hundreds of thousands of movements per second. It is obviously wasteful to have a machine at these superhuman speeds unless the information it is to work upon can be made available to it at a relatively comparable speed. (Thompson and Standingford 1947, section D.I)

Notable, too, was their speculation on the potentially transformative effect computers could have on the way business might be organized. The report continues with notes on coding and binary, input media such as punched cards, punched tape, magnetic wire and magnetic tape, and forms of memory, such as acoustic delay line, selectron, magnetic drum and sound track recording, later described in the appendices. The report includes a detailed list of contacts made in the USA.

Prior to the trip, on February 25, 1947, they contacted Herman Heine Goldstine (1913–2004), the mathematician and computer scientist who helped in the building of early computers and was involved in developing the ENIAC (Electronic Numerical Integrator and Computer) at the Moore School, University of Pennsylvania. Goldstine suggested to them a number of companies, labs and people to visit during the trip. These included Howard Hathaway Aiken (1900–1973), who had built one of the first electromechanical computers at Harvard University, and a number of US companies that eventually became computer manufacturers, such as RCA, IBM, NCR and Burroughs Corporation. In their first meeting at the Institute for Advanced Studies of Princeton University, Thompson and Standingford discussed the possibilities of office applications for computers. Goldstine himself had not thought about a computer for business applications yet. This is not surprising as computers at the time were thought of as machines to be used for military and scientific calculations. He reacted to the idea in an enthusiastic way, outlining possible solutions and inviting them back at the end of their trip to discuss these matters further.

During their North American trip, Thompson and Standingford visited the Prudential Life Insurance Company of America in Newark, where they met Edmund Callis Berkeley (1909–1988), in charge of the section of the O&M division. Berkeley was one of the officers who operated the Harvard Mark I (a primarily mechanical relay-driven computer) under Aiken during the Second World War and was also the founder of the Eastern Association for Computing Machinery (EACM) as early as 1947; it eventually became the ACM (Aker 2007), one of today's most important societies for computing. Moreover, during his tenure at the O&M department of Prudential, Berkeley planned to build a computer. The project eventually became the UNIVAC I (Universal Automatic Computer I). The first model was delivered to the US Bureau of Census in March 1951. The report provides detailed information about Berkeley's plans:

Their project is to build a machine that will carry out the premium billing of their millions of policy holders, a job that at present occupies a staff of some 300 clerks. They anticipate that this work can be done with less than 20 percent of the staff in about two days per month. The remainder of the machine's time will be given up to actuarial calculations. They are proceeding with detailed plans, and Dr. Berkeley is confident that they will have a machine installed within about two years. When they have established premium billing, they propose to continue their investigation with a view to putting other routine work on the machine, including the writing of contracts for which they have some 200 standard clauses from which to select. (Thompson and Standingford 1947, section D.VI)

Back in the UK, Standingford and Thompson visited the Cambridge Mathematical Laboratory, where they met physicist Douglas Rayner Hartree (1897–1958), well-regarded Cambridge mathematician and physicist involved in the early application of digital computers, and were introduced to Maurice Vincent Wilkes (1913–2010), who was building the EDSAC (Electronic Delay Storage Automatic Calculator). They were most impressed by advances in the technology and by what they could see of the EDSAC's construction in progress. Both Wilkes and Hartree were enthusiastic and interested in applying the Cambridge machine to clerical jobs in collaboration with Lyons.

In the report they prepared after the trip, Thompson and Standingford outlined possible applications of electronic machines to office work, such as payrolls and sales invoicing. Another possible application was letter writing, i.e., an automated system with codes for customers, and variations of standard salutations, paragraphs, and closures (Thompson and Standingford 1947, section D.V). This idea echoes Berkeley's plan for Prudential to use computers to write contracts.

In order to get a working computer as soon as possible it was decided that an adapted copy of the EDSAC computer would provide a sound basis for Lyons' computer. With this in mind they decided to fund part of the development of the machine. In exchange, when the EDSAC proved to be working, Cambridge would help Lyons to build its own business computer.

5. From EDSAC to LEO

In November 1947, the Lyons Board donated £2,500¹⁶ and the services of an assistant to the Cambridge computer project. In exchange, they were allowed to copy and adapt the computer design—conceived for scientific

16 The *LEO Chronicle (1947–1954)* (Lyons & Co. [1954]) reports £2,500 whereas other sources report £3,000. The *LEO Chronicle* is available at the Modern Records Centre, Warwick University in the Papers of John Simmons, MRC MSS.363/54/11 and online.

purposes—for their own business applications.¹⁷ But Lyons would not start its own project until Cambridge could demonstrate a working application on its EDSAC.

It is possible to trace Lyons' O&M techniques also later in the design and development of their computers, in the improvement of operating and programming procedures, progressively diverging in both fields (engineering and programming) from the practice in place of what had been their model at Cambridge.

The EDSAC executed its first program on Friday May 6, 1949; regular use began in early 1950 and it was closed down in July 1956. The programming scheme, devised by David John Wheeler (1927–2004), was based on a library of closed subroutines that could be reused. A subroutine was a self-standing routine to carry out a mathematical function, such as division, constructed in such a way that when it was entered from another sequence it returned control after its execution to the original sequence. Subroutines could be of two kinds: open and closed. Open subroutines were incorporated at each point in the program where they were needed, whereas closed subroutines appeared only once in a program and were called by a special transfer-of-control sequence at each point in the program where they were needed (Campbell-Kelly 1980, 17). The practice at Cambridge was to select the appropriate closed subroutines for a given application and to write what was called a “Master Routine” to enter in turn the various closed subroutines that had been picked up. Open subroutines were rarely used at Cambridge with the exception of course of these Master Routines. It is important to avoid confusion: the open Master Routines at Cambridge are not the Master Routine Operating System (OS) later devised for LEO III, but it is possible that the OS' name was inspired by this practice.

The EDSAC library of subroutines had three main advantages: reducing the lines of code, providing more efficient subroutines written by experts, and offering major reliability because they were thoroughly tested beforehand. An EDSAC Programme Sheet documented each subroutine and helped unskilled programmers to use them optimally (Campbell-Kelly 2019). Initially users operated the EDSAC themselves but in January 1950 the Laboratory established a formal computing service, composed of an operation manager, a senior and a junior operator, and an engineer who fixed the machine when it had issues. This meant that users could no longer operate the machine themselves.¹⁸

17 See *Electric Machines: Report of a Visit by Mr Booth and Others to Cambridge*, November 11, 1947, in Papers of John Simmons, Modern Records Centre Warwick University, MSS.363/S4/2/1/41, Pinkerton et al. (1992), and Bird (1994, 38–9).

18 There were two queues for these jobs, with a visual color code tag: yellow for experimental programs (that needed debugging and were given priority) and blue for production programs. Each “EDSAC tape ticket” included a set of instructions: the amount

Lyons established their own team of engineers to begin the construction of their machine: it was led by John Maurice McClean Pinkerton (1919–1997), a young Cambridge University researcher at the Cavendish Laboratory, the University's Department of Physics.¹⁹

System design and program development for LEO proceeded in parallel with physical construction. The programming team was built up and it was guided by David Tresman Caminer (1915–2008). Many programmers had already been Lyons' employees in the Systems Research Office before joining the LEO team. Caminer himself had been manager of the Systems Research Office before being seconded to the LEO group. Thomas R. Thompson led the whole LEO team. At the end of 1947 Derek Hemy (1920–2000), who was one of the Lyons' O&M experts, began the study of a clerical application, a payroll to be tested on the EDSAC in collaboration with David Wheeler.²⁰ Wheeler had volunteered as an undergraduate to help build the EDSAC in 1946. The following year he began his PhD under Wilkes and his thesis (Wheeler 1951) formed the basis of *The Preparation of Programs for an Electronic Digital Computer*, the first and influential textbook on programming, also known as "Wilkes, Wheeler, and Gill" or WWG, from the surnames of its authors (Wilkes, Wheeler, and Gill 1951).

They chose a payroll because it would be a good test for a series of reasons. First, the reliability of the machine and the obtained results could be checked in detail against the manual computations produced in parallel by the Lyons' Wages Office. Second, a payroll involves permanent data, which doesn't change weekly (such as the name of the employees and their National Insurance number), and variable data (i.e., the hours each employee worked weekly, exceptions like bonuses, and so on). Third, Lyons could evaluate and establish which clerical savings could be made by using the computer in comparison with the manual calculations (this was key to Lyons as shown by the forms devised at a later stage). Fourth, a payroll has specific time requirements, from the availability of data to be used on a given day, to the deadline for pay slips, on a regular weekly time cycle.

of memory needed, the expected run time, and the expected output. The users could indicate whether they wanted to be present during the run or specify what to do in the event of an abnormal termination. In 1952 red-tagged tapes were added and given priority over the yellow tapes and they consisted of new library subroutines and machine test programs (Campbell-Kelly 2019, 126).

- 19 Pinkerton had read Natural Sciences at Cambridge's Trinity College, and later worked on radars at the Air Ministry Research Establishment in Swanage. He then went to the Telecommunication Research Centre (TRE) in Malvern, where he met Wilkes. He was hired in December 1948 on the recommendation of Wilkes. See Bird (1994, 44–45) and M. Wilkes, private communication to J. Simmons, December 1, 1948, Peter Bird's private collection.
- 20 See *Report on Payroll Preparation by Electronic Computer*, January 16, 1948, Papers of John Simmons, Modern Records Centre Warwick University, MSS.363/54/2/1/38.

As a result of this test, and following O&M practice (Mogensen 1932), Lyons decided to follow EDSAC's design with no changes until they fully understood why each solution had been chosen and why anything should be changed. Hemy's payroll program indicated the necessity to double the size of EDSAC's memory, and to improve and speed up input/output peripherals.²¹ This payroll was a virtual one, never planned to be used in reality, but intended to both learn and to demonstrate its feasibility.

6. Flowcharts

The use of the diagrammatic notation for the analysis of the operation of a computer is considered to have been introduced in the field of computing by Goldstine and von Neumann (1947) in their *Planning and Coding of Problems for an Electronic Computing Instrument*.²² In O&M process charts (see examples in Figures 1 and 2) had existed since the 1920s, used by the Methods people to organize and improve the processes in factories and offices. As mentioned in section 4 Lyons had direct contacts with Goldstine in May 1947, on the occasion of their fact-finding trip by Standingford and Thompson. The flowcharts used in the programming practice and in general at LEO were both the result of their experience in O&M and the Goldstine and von Neumann approach to planning and coding. The flowcharting method was also adopted at the Eckert–Mauchly Computer Corporation, where the UNIVAC computer was developed, and where Grace Hopper (1906–1992) and Betty Holberton (1917–2001) used them in a programming course, referencing Von Neumann and Goldstine. Flowcharts can also be found in the documentation for a program developed for the earliest versions of the ACE computer, designed by Alan Turing (1912–1954) at the National Physical Laboratory in Teddington, London.

Whereas the EDSAC users relied for much of their work on a library of sub-routines for mathematical functions, held on tape and ready to be copied for incorporation into user programs, at Lyons the design of a program began with the analysis of an application and the development of a strategic master flowchart. The program was only then devised in a sequence of open sub-routines, which corresponded to what Lyons called "stages". An example of a flowchart of a routine and its coding is shown in Figures 3a and 3b respectively.

21 *LEO (report no. 6)*, January 31, 1950, Papers of John Simmons, Modern Records Centre Warwick University, MSS.363/S4/2/1/67.

22 See Priestley (2020). A historical and sociological analysis of the *The Multiple Meanings of a Flowchart* in computing can be found in Ensmenger (2016), where Ensmenger interprets flowcharts as boundary objects, blueprints, and documentation tools, within their limits and failures. Morris and Gotel (2006) is a reconstruction of the success, formalization, and demise of flowcharts in computing.

As already noted, the approaches of Cambridge and Lyons tended to diverge. Cambridge built a comprehensive library of closed subroutines to carry out mathematical functions, whereas at Lyons they thought that the routines of business and commercial work were too full of variations and exceptions to have this kind of approach, because it would be too costly in operating time and would have led to use of extra storage. The open subroutines approach was regarded as more suitable and efficient; however, a few closed subroutines were used, in particular for input and output operations (Bird 1994, 57).

Derek Hemy visited Cambridge for two weeks in May 1948 studying programming methods. Back at Lyons, Hemy discussed with David Caminer his idea of devising a form of flow charting and was encouraged by him. David Wheeler visited Lyons in the summer of 1948 and thereafter shared his programming expertise. Hemy, Caminer, and Wheeler began to start thinking about the requirements for business programming and how it differed from mathematical applications. When the small group began to work on their payroll application to test on the EDSAC the first divergence in approaches appeared: while Hemy advocated for the use of flowcharting techniques, Wheeler believed they were not necessary tools for programming (Bird 1994, 50–52). Hemy, retrospectively, defined the purposes and uses of flowcharts as:

The first function was to simplify the analysis of a process into logical steps, each of which might represent several orders, as an aid to structuring a program. Second, such a chart, by showing where and under what condition choices were to be made and the processing that would result, was helping in checking the logic of a program before it was written and could lead to better coding. Third, when a program had been written the chart would provide a guide and index to the coded orders. (Bird 1994, 51–52)

The scarce use of flowcharts in the EDSAC practice is confirmed by a note by Wilkes, in his papers held at the Cambridge University Archives; the note is attached to a folder that contains two programs required for the calculations of serial correlation coefficients and it declares:

These programs were written for a specific application but they also formed an experiment in programming methodology. They are both built around a central counting subroutine. I was, at the time, testing a theory that this form of structuring would make programs easy to write, understand, and debug. I was soon disillusioned. These programs are among the few that I have ever written that needed flow diagrams for their description.²³

23 Note by Maurice Wilkes, July 17, 1980, Cambridge University Archives, Comp A5 1*.

In his recollections, Wheeler (1992, 36) stated:

The use of program pads and the breaking down of a program into sub-routines meant that flowcharts were never used as programming aids for the EDSAC.

However, it is interesting to note that in his paper published in the *Report of a Conference on High Speed Automatic Calculating-Machines*, which took place in Cambridge on June 22–25, 1949, Wheeler defined a program as a flowchart:

A PROGRAMME²⁴ is a flow chart showing the operations (in block form) corresponding to the action of the calculator during the solution of the problem;

A ROUTINE is the programme written in the detailed code of a particular calculating machine;

A SUB-ROUTINE is the coded form of an element in a routine (e.g., square root, input and binary decimal conversion, etc.). (Wheeler 1950, capitalized words in the original)

This definition of “programme” might have been influenced by the internship he had at Lyons in the summer of 1948. Interestingly, his definition of programming in his PhD thesis (1951) is slightly different and doesn’t involve flowcharts any more:

The process of drawing up a set of orders that will enable the machine to do a certain calculation is known as programming, and the set of orders is called the programme. (Wheeler 1951, 7, underlined words in the original)

Even within the EDSAC team the definition of what a program was at this stage was still evolving. In the EDSAC short film (EDSAC Film 1951) produced at Cambridge for the first Eastern Joint Computer Conference held in Philadelphia in December 1951²⁵ we can find a slightly different definition of program, which connects the sequencing of the orders with the requirement to be appropriate for the machine:

Before a problem can be solved on an automatic digital calculating machine such as the EDSAC it must be broken down into a sequence of elementary steps—multiplications, additions, etc. When these are

24 “Programme” was the British spelling of the word “program” in the particular case of a computer program. This spelling progressively disappeared during the 1950s and 1960s.

25 The Joint Computer Conferences were held under various names between 1951 and 1987 in the USA. Originally, they were composed of two events every year: the Western Joint Computer Conference (WJCC) held annually in the western USA, and its counterpart the Eastern Joint Computer Conference (EJCC). Both conferences were sponsored by the National Joint Computer Committee (NJCC), composed of the ACM, the American Institute of Electrical Engineers (AIEE) Committee on Computing Devices, and the Institute of Radio Engineers (IRE) Professional Group on Electronic Computers.

expressed in a form which can be accepted by the machine the result is known as the programme.²⁶

The original 1949 Cambridge conference proceedings, where Wheeler defines a program as a flowchart, included also several papers using flow diagrams (Eggink 1950; Renwick 1950), showing how they were already diffused in the UK, at least as a representation technique also in the EDSAC team. “The E.D.S.A.C. Demonstration” paper by William Renwick (1924–1971), prepared by Beatrice Helen Worsley (1922–1972), used notes, flow diagrams, and annotations to “render the actual routines used in the demonstration intelligible” (Renwick 1950). In his paper “Checking a Large Routine” Turing (1950) also used a flowchart as a communication tool, stating:

Unfortunately, there is no coding system sufficiently generally known to justify giving the routine for this process in full, but the flow diagram given in Fig. 1, will be sufficient for illustration. (Turing 1950, 67)

Wheeler’s PhD thesis *Automatic Computing with the EDSAC* doesn’t mention the flowcharting technique any more, but uses a flowchart to illustrate a subroutine that tested numbers for primality on the EDSAC. Wheeler commented:

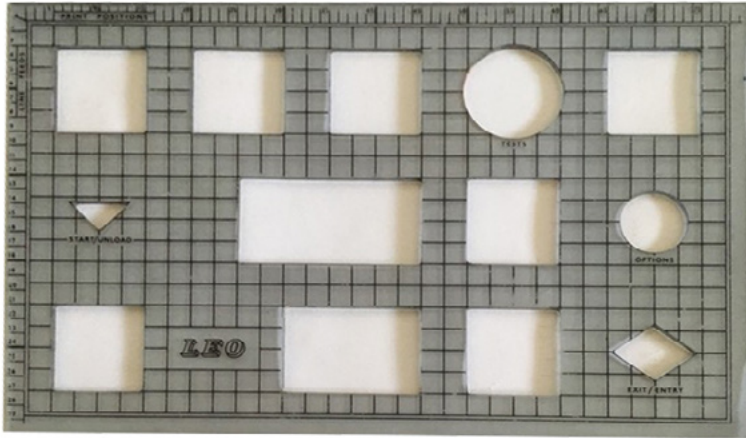
A diagram illustrating the action of the subroutine ... in the form of a flow diagram originally described by von Neumann. (Wheeler 1951, 107)

Both Wheeler and Wilkes later criticized the use of flowcharts in programming, except as a representation of a written program to aid a third party reading the program, and the Wilkes, Wheeler, and Gill book (based on Wheeler’s thesis) does not mention or show flowcharts at all (Wilkes, Wheeler, and Gill 1951). Wheeler, interviewed by William Aspray in 1987, remembered:

We certainly knew, for example, about von Neumann’s flow diagrams. These were never used, except possibly as an explanation of what had been done. We discovered that if you basically took the problem, decomposed it into subroutines, and put these under the control of the master routine, this more or less forced you to think modularly and enabled you to do all these programs without going to the flow diagrams, which in my own personal opinion contributed more to bad programming than almost anything else. This is not to say that they don’t have their place, but if they’re used as a substitute for thought in the preparation of your program, they don’t work.²⁷

26 The film, made in 1951 with a commentary by Maurice Wilkes added in 1976, is available online (EDSAC Film 1951). In the film, some of the EDSAC team members act, and in particular Stanley Gill (1926–1975, one of the authors of WWG) appears as a programmer.

27 Oral History of David J. Wheeler, conducted by William Aspray (Wheeler 1987), available online.



[Fig. 4] LEO stencil for drawing flowcharts (Source: Courtesy of Graham Briscoe).

Despite Wheeler and Wilkes' little use of flowcharts for the EDSAC, they were employed in many contexts in both the USA and UK, as shown in Morris and Gotel (2011). Usually the process of programming was split in two stages: a first that consisted of converting the numerical methods in a series of steps, expressed in the form of a flow diagram; and a second, often called "coding," which consisted of writing down the actual orders.

The use of flowcharts remained unstandardized until the mid-1960s, when standards were created in Europe and the USA. In Europe a technical committee of the European Computer Manufacturers Association met for the first time in 1960 to prepare a standard for "Flow Charts." It included representatives of Compagnie des Machines Bull, Dutch Electrologica, Ferranti, IBM, ICT, ITT Europe, Olivetti, SEA, Siemens, Sperry Rand International Corp., and Telefunken, in collaboration with the American Standards Association (ASA), the Deutscher Normen Ausschuss (DNA), and the International Organization for Standardization (ISO). The Standard ECMA-4 on "Flow Charts" was published in November 1964 (ECMA 1964). In July 1965 the American Standards Association published a working paper in the *Communications of the ACM*. The working document was produced by the ASA sub-committee Sectional Committee X3, Computers and Information Processing, to develop a US standard (Gorn 1965).

7. Conclusions

The different working environments of LEO and EDSAC required different machines. EDSAC was used by a multiplicity of academics and researchers, unlikely at that time to have previous programming knowledge or skills, who

had to run complex, usually one-off, mathematical computations.²⁸ LEO I was required to run large business applications, involving large programs and high volumes of transactions, to a tight time-critical schedule, requiring major needs of memory and different input/output devices. In business computing, a large number of simple calculations had to be accomplished in the shortest possible time, as opposed to mathematical and scientific work, which were typically required to perform a small number of highly complex calculations. Commercial computing led to different problems, related to the volume of data needed to be input, results to be output, and files recorded. Data entry was time consuming while calculations were simple. For each application a large program (or sets of programs) had to be run repeatedly for each transaction or batch of data.

Programs for clerical jobs tended to be much bigger and there were two main reasons for this. First, once the data was loaded in the memory, it was more economical to carry out as many operations as possible on it. The program would then be written to cover a series of interrelated routines rather than just one, making it longer. Second, the requirements of such jobs were less logically structured than mathematical ones, with complexity originating in commercial and legal regulations that led to many possible variations and, indeed, anomalies (Thompson 1954, 90). Program design and correctness was also a different concern in the two environments.

Program correctness was pursued by the analytic decomposition of each program in parts, starting from their main function (specification), and their reconstruction through detailed flowcharts. Flowcharts (and then code) were carefully checked by different colleagues beforehand.

Early business computers were not just the adaptation of early scientific computers modified for business purposes: they were also the result of the influence of the Scientific Management and O&M practices in place since the 1920s. In particular, the role of Lyons in the British O&M movement and their well-known use of flowcharts in business administration influenced the development of programming techniques and checking routines at LEO. While in the university setting of Cambridge the flowcharting method was not regarded as of critical importance, in LEO it represented a common discipline, a tool for understanding complex business procedures, a planning aid, and a way to test and check their correctness before running programs on the machine.

This publication is part of the JDC2023-051119-I grant, financed by MCIU/AEI/10.13039/501100011033 and by the FSE+. The author would like to acknowledge: the PRO-GRAMme project at the University of Lille (ANR-17-CE38-0003-01); the Association for Information Technology Trust - David Tresman Caminer Scholarship for the History of Computing (2016-2019); the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support

28 The evolution of the computing practice on the EDSAC is described in detail in Campbell-Kelly (2019).

and hospitality during the programme *Modern History of Mathematics*, where work on this paper was undertaken. This work was supported by EPSRC grant EP/Z000580/1. The author would also like to thank Arianna Borrelli and Helena Durnová for making this volume possible; Hilary Caminer, Frank Land, John Daines, Graham Briscoe, and LEO Computers Society; Lisa McGerty and Kate Stockwell, at the Centre for Computing History, Cambridge; Liesbeth De Mol and the PROGRAMme community; Andrew Herbert, Jacqui Garrad and the EDSAC Replica Project team at TNMOC, Bletchley Park; Daniele Cozzoli and the History of Science and Modern Research Group, Universitat Pompeu Fabra, Barcelona; Balbir Barn, Michele Bottone, Tony Bryant, Martin Campbell-Kelly, Snezana Lawrence, Giuseppe Primiero, Mark Priestley and Brian Randell.

References

- Agar, Jon. 2003. *The Government Machine: A Revolutionary History of the Computer*. Cambridge, MA and London: MIT Press.
- Akera, Atsushi. 2007. "Edmund Berkeley and the Origins of ACM." *Communications of the ACM* 50, no. 5: 30–35.
- Aris, John B. B. 2000. "Inventing Systems Engineering." *IEEE Annals of the History of Computing* 22, no. 3: 4–15.
- ASME. 1947. *Operation and Flow Process Charts*. New York: The American Society of Mechanical Engineers. Accessed October 2024. <https://babel.hathitrust.org/cgi/pt?id=mdp.39015039876274&view=1up&seq=3>.
- Bird, Peter J. 1994. *LEO: The First Business Computer*. Wokingham: Hasler Publishing Limited.
- . 2000. *The First Food Empire: A History of J. Lyons & Co*. Chichester: Phillimore & Co. Ltd.
- Blaschke, Estelle. 2016. "Installed for Your Protection: Mikrofilm als Medium der Bürokratie." In *Medien der Bürokratie*, edited by Friedrich Balke, Bernhard Siegert, and Joseph Vogl, 151–62. Paderborn: Wilhelm Fink Verlag.
- Bryant, Anthony, and Frank Land. 2020. "A 'Conversation' between Frank Land (FL) and Antony Bryant (AB): PART I." *Journal of Information Technology* 35, no. 4: 361–75.
- Caminer, David T., John B. B. Aris, Peter Hermon, and Frank Land. 1996. *User-Driven Innovation. The World's First Business Computer*. London: McGraw-Hill.
- Caminer, David T., John B. B. Aris, Peter Hermon, and Frank Land. 1998. *LEO: The Incredible Story of the World's First Business Computer*. New York: McGraw-Hill.
- Campbell-Kelly, Martin. 1980. "Programming the EDSAC: Early Programming Activity at the University of Cambridge." *Annals of the History of Computing* 2, no. 1: 7–36.
- . 2019. "The Evolution of Digital Computing Practice on the Cambridge University EDSAC, 1949–1951". In *Exploring the Early Digital*, edited by Thomas Haigh, 117–34. Cham: Springer.
- Cannons, Harry G. T. 1920. *Bibliography of Industrial Efficiency and Factory Management*. London and New York: George Routledge & Sons, Ltd and E. P. Dutton & Co.
- Chandler, Alfred D. Jr. 1962. *Strategy and Structure: Chapters in the History of the American Industrial Enterprise*. Cambridge, MA: MIT Press.
- . 1977. *The Visible Hand: The Managerial Revolution in American Business*. Cambridge, MA and London: The Belknap Press of Harvard University Press.
- ECMA. 1964. *ECMA-4 Standard for Flow Charts*. Geneva.
- EDSAC film. 1951. *The EDSAC Film, with a Commentary by Maurice Wilkes (1976)*. Accessed October 2024. <https://sms.cam.ac.uk/media/739532>.
- Eggink, H. 1950. "The Programming of Supersonic Nozzle Flow." In *Report of a Conference on High Speed Automatic Calculating-Machines, June 22–25, 1949*, 47–49. Cambridge: Cambridge University Mathematical Laboratory with the Co-operation of the Ministry of Supply.
- Emerson, Harrington. 1909. *Efficiency as a Basis for Operation and Wages*. New York: The Engineering Magazine.
- Ensmenger, Nathan. 2016. "The Multiple Meanings of a Flowchart." *Information & Culture* 51, no. 3: 321–51.

- Ferry, Georgina. 1985. "John Simmons (BC. 1916), Pioneer of business systems—"The Father of the Office Computer." Accessed October 2024. <https://www.oldbrightonians.com/news/notable-obs/business/243-john-simmons-bc-1916.html>.
- . 2004. *A Computer Called LEO: Lyons Teashops and the World's First Office Computer*. London: Harper Perennial.
- Gantt, Henry L. 1910. *Work, Wages and Profits: Their Influence on the Cost of Living*. New York: The Engineering Magazine.
- Gibson, A. 1991. *The Institute of Administrative Management – Seventy-Five Innovative Years*. Modern Records Centre, Warwick University, MSS.337/4/66/1.
- Gilbreth, Frank B. 1907. *Field System*. New York and Chicago: M. C. Clark Publishing Co.
- . 1909. *Bricklaying System*. New York and Chicago: M. C. Clark Publishing Co.
- . 1911. *Motion Study: A Method for Increasing the Efficiency of the Workman*. New York: D. Van Nostrand Company.
- . 1912. *Primer of Scientific Management*. New York: D. Van Nostrand Company.
- , and Lillian M. Gilbreth. 1917. *Applied Motion Study: A Collection of Papers on the Efficient Method to Industrial Preparedness*. New York: Sturgis & Walton Company.
- , and Lillian M. Gilbreth. 1921. "Process Charts. First Steps in Finding the One Best Way to Do Work." *Transactions of the American Society of Mechanical Engineers* 43: 1029–43.
- Goldstine, Herman H., and John von Neumann. 1947. *Planning and Coding of Problems for an Electronic Computing Instrument*. Princeton: Institute for Advanced Study.
- Gorn, Saul. 1965. "Conventions for the Use of Symbols in the Preparation of Flowcharts for Information Processing Systems." *Communications of the ACM* 8, no. 7: 439–40.
- J. Lyons & Co. [1954] *The LEO Chronicle (1947–1954)*. Modern Records Centre, Warwick University, Papers of John Simmons, MRC MSS.363/54/11. Accessed October 2024. <https://cdm21047.contentdm.oclc.org/digital/collection/leo/id/242> and Centre for Computing History CCH CMLEO DC/WF/LCR/1. Transcription at [http://www.computinghistory.org.uk/det/55876/55876-LEO-Chronicle-\(with-1955-61-extension\)](http://www.computinghistory.org.uk/det/55876/55876-LEO-Chronicle-(with-1955-61-extension)).
- Kreis, Steven. 1992. "The Diffusion of Scientific Management: The Bedaux Company in America and Britain, 1926–1945." In *A Mental Revolution: Scientific Management Since Taylor*, edited by D. Nelson, 156–74. Columbus: The Ohio State University Press.
- Mills, Geoffrey, and Oliver Standingford. 1949. *Office Organization and Method*. London: Sir Isaac Pitman & Sons, Ltd.
- Mogensen, Allan H. 1932. *Common Sense Applied to Motion and Time Study*. New York and London: McGraw-Hill.
- Mori, Elisabetta. 2022. "Pursuing Efficiency in Business: The Technological and Commercial Development of LEO Computers." PhD Thesis. Department of Computer Science, Middlesex University, London.
- Morris, Stephen J., and Orlena C. Z. Gotel. 2006. "Flow Diagrams: Rise and Fall of the First Software Engineering Notation." In *Diagrammatic Representation and Inference*, edited by D. Barker-Plummer, R. Cox, and N. Swoboda, 130–144. Diagrams 2006. Lecture Notes in Computer Science 4045. Berlin and Heidelberg: Springer-Verlag.
- Morris, Stephen J., and Orlena C. Z. Gotel. 2011. "The Role of Flow Charts in the Early Automation of Applied Mathematics." *BSHM Bulletin: Journal of the British Society for the History of Mathematics* 26: 44–52.
- Nelson, Daniel, ed. 1992. *A Mental Revolution: Scientific Management Since Taylor*. Columbus: The Ohio State University Press.
- Pinkerton, John M. M., Derek Hemy, and Ernest H. Lenaerts. 1992. "The Influence of the Cambridge Mathematical Laboratory on the LEO Project." *IEEE Annals of the History of Computing* 14, no. 4: 41–48.
- Price, Brian. 1992. "Frank and Lillian Gilbreth and the Motion Study Controversy, 1907–1930." In *A Mental Revolution. Scientific Management Since Taylor*, edited by Daniel Nelson, 58–76. Columbus: The Ohio State University Press.
- Priestley, Mark. 2020. "Flow Diagrams, Assertions, and Formal Methods." In *Formal Methods: FM 2019 International Workshops, Revised Selected Papers, Part II*: 15–34. London: Springer.

- Renwick, William. 1950. "The E.D.S.A.C. Demonstration." In *Report of a Conference on High Speed Automatic Calculating-Machines, 22-25 June 1949*, 12-16. Cambridge: Cambridge University Mathematical Laboratory with the Co-operation of the Ministry of Supply.
- Simmons, John R. M. 1962. *LEO and the Managers*. London: Macdonald & Co.
- Taylor, Frederick W. 1911. *The Principles of Scientific Management*. New York and London: Harper & Brothers.
- Thompson, Thomas R. 1954. "Special Requirements for Commercial or Administrative Applications." In *Proceedings of a Symposium Held at the National Physical Laboratory on March 25, 26, 27 & 28, 1953*, 85-101. London: Her Majesty's Stationery Office.
- Thompson, Thomas R., and Oliver W. Standingford. 1947. "American Tour Report, May & June 1947." London: J. Lyons & Co. Accessed October 2024. <http://www.computinghistory.org.uk/sec/63008/American-Tour-T-R-Thompson-and-O-W-Standingford-May-and-June-1947>.
- Turing, Alan M. 1950. "Checking a Large Routine." In *Report of a Conference on High Speed Automatic Calculating-Machines, 22-25 June 1949*, 67-68. Cambridge: Cambridge University Mathematical Laboratory with the Co-operation of the Ministry of Supply.
- Wheeler, David J. 1950. "Planning the Use of a Paper Library." In *Report of a Conference on High Speed Automatic Calculating-Machines, 22-25 June 1949*, 36-40. Cambridge: Cambridge University Mathematical Laboratory with the Co-operation of the Ministry of Supply.
- . 1951. "Automatic Computing with the EDSAC." PhD Thesis, Cambridge: University of Cambridge.
- . 1987. *Oral History of David J. Wheeler (OH 132) conducted by William Aspray on 14th May 1987*. Princeton, NJ, Charles Babbage Institute, The Center for the History of Information Processing, University of Minnesota, Minneapolis. Accessed October 2024. <https://conservancy.umn.edu/items/1764b7d6-5036-4110-ac8a-7e7224138ed5>.
- . 1992. "The EDSAC Programming Systems." *IEEE Annals of the History of Computing* 14, no. 4: 34-40.
- Wilkes, Maurice V., David J. Wheeler, and Stanley Gill. 1951. *The Preparation of Programs for an Electronic Digital Computer*. Reading, MA: Addison-Wesley.
- Yates, JoAnne. 1989. *Control Through Communication. The Rise of System in American Management*. Baltimore and London: The Johns Hopkins University Press.

CHARLES BABBAGE

DIVISION OF LABOR

SYMBOLICAL ALGEBRA

DIFFERENTIAL ANALYSER

PROGRAMMING

The Division of Mental Labor in Computing Practices: Presuppositions, Advances, Biases

Marie-José Durand-Richard

When Charles Babbage (1791–1871) designed the Difference Engine and the Analytical Engine, he maintained they illustrated how the principle of the division of labor can govern mental operations of the mind as well as material operations in industry. In his view, such a rational organization of work was a guarantee of harmony for a hierarchical society guided by learned men. Yet, Babbage and the early history of computer science missed the analysis of the conditions of production for the development of computing devices which followed more closely socio-economic needs during the 19th and the 20th centuries. Exploring nature took place rather through the production of analog devices, culminating with the differential analyzer, while computations organizing social activities used increasingly sophisticated desk calculators. All these devices produced new practices that would be embodied in electronic

high-speed machines. If the division of labor was emphasized in the specialization of scientists (mathematicians, physicists, engineers), strong local collaborations were needed between them to get machines up and running. Computers were elaborated in a context where, after the two world wars, new relationships between science, industry and politics (essentially defense) were established and maintained, and where scientists were more dependent on economies than Babbage could imagine. Electronic computers led to a very fine decomposition of computations, so that the division of labor seemed to be extreme at the operating level. But it appeared that computers would be useless without programming, which led to new operations between these elementary units, and that programming languages would be adapted to specific needs. At the socio-economic level, as computers became commercial artefacts, the invisibility of automatic processes induced users to become passive facing computers and their place in society. In any case, the complexity of these interactions needs to be analyzed so that all actors remain fully aware of the wide range of meanings attached to automatic computing in human affairs.

The starting point of my paper is Charles Babbage (1791–1871), and the statement he made in his book *On the Economy of Machinery and Manufactures* (1832). He considered that the division of labor, as it was practiced in manufactures from the Industrial Revolution (1750–1830) and analyzed by Adam Smith in 1776 in *An Inquiry into the Nature and Causes of the Wealth of Nations*,

could be applied in the same manner to mental work, more precisely to mechanize computing as exemplified by his Difference Engine. It is tempting to read Babbage in a retrospective way, and his later plans for the Analytical Engine are often interpreted according to a Whiggish view of history, presenting him as a pioneer of computer science (Hyman 1983). Today, historians tend to insist more on the importance of integrating scientific works into their contextual framework, notably for the history of mathematics (Dauben and Scriba 2002) and of computer science (De Mol and Bullynck 2018). This paper intends first to examine Babbage's interest in the division of labor from this contextual point of view. What he said on mental work about the Difference Engine was directly inspired by specific views on algebra, collectively elaborated in the context of the reform of Cambridge mathematics, so as to establish it as a science. It will be seen that this viewpoint focused on a radical separation between the process of operations and their numerical results. Babbage founded the efficiency of mechanizing on a strict hierarchy between the two of them and focused on the first. Yet, due to his background at Cambridge and his social position as a "gentleman amateur" (Morrell and Thackray 1981), he failed to analyze what his relationships with the industrial world should have been in the context of the Industrial Revolution, so as to make possible the construction of such a calculating machine. In other words, he viewed machines as neutral, and did not consider their conditions of production (Pestre 2006), including the material and organizational structure to support them, the effective needs for numerical results, and their interactions with the process of computing itself. This paper will outline and contextualize the complex developments that took place between Babbage's time and the period after the Second World War. If these developments are often seen *a posteriori* as implementing Babbage's plan on mechanizing computational work, they also show the limits of Babbage's ideas on the division of mental labor.

Despite recent analyses, the popular history of computing has often developed a retrospective view of the field, and considered that nothing happened between Babbage and Alan Turing (1912–1954). This was a wrong appreciation of the role of Turing in the advent of digital electronic computers (Daylight 2015), belonging to a historical trend which only accounted for what happened in the mechanization of digital computing (Trogemann, Nitussov, and Ernst 2001; Williams 1985). However, as I will discuss more in detail later on, during this period the social and technical environment of mathematics evolved significantly. Engineers, physicists and applied mathematicians were deeply committed to the structuring of new spaces and artefacts throughout industrial Europe and beyond (Durand-Richard 2022). In particular, in order to solve mathematical problems of integration in experimental situations, they developed a long line of analog instruments and machines, from planimeters to harmonic and differential analyzers. The conditions of production of these devices had radically changed since Babbage's endeavor, as they accompanied

the development of transnational engineering culture and experimental physics. As their users were scientists, their construction and running mobilized various researchers and workers, so that in spite of disciplinary specializations, the division of labor was not so strict as Babbage contemplated it.

The Second World War substantially reinforced this crucial increase of needs for numerical computations in various industrial and scientific fields, from ballistics to armament production, and radically transformed their development framework. The construction of new computing machines, from electromechanical to electronic ones, was heavily supported by huge institutional networks, joining universities, industry, and defense. The division of labor in industry and computing laboratories—whether manual, mental, or organizational—was reinforced by a severe requirement for secrecy, and regulated the complete automatization of digital computing machines.

Both these computational needs and the way they were dealt with did not disappear after the Second World War, both because of the Cold War and of the later extension to commercial applications in the 1950s. Yet, computer scientists discovered that mechanizing computations required more than isolating their simple steps. Defining elementary operations and specifying their succession were not enough for programming a computer. The introduction of stored-program machines doubly changed the game. It appeared that the whole process of computations could not be strictly reduced to the division of mental labor, while the complete automatization of machines tended to radically hide the meaning of computations to the users, an issue that could not be perceived as long as users were solely scientists.

1. Babbage's Commitment to Science and Economics

From the start of his scientific commitment in 1813, Babbage was a leading member of a group of algebraists in Cambridge, which in the 1830s will become the core of an important network with multiple ramifications (Cannon 1964). This network clearly understood what was at stake with the Industrial Revolution, both for science and national institutions (Durand-Richard 2000). The period was highly disturbed, with riots and the destruction of machines in industrial cities. With the intention of preventing such a political upheaval as the French Revolution, and to warrant stability in English society, this network worked to integrate the empirical experience gained in industry into the canons of science at political and scientific levels. Many members of this network, politically close to Whigs and Radicals, were among “the Gentlemen of Science” who created and managed the British Association for the Advancement of Science during its first 20 years, so as to maintain social

cohesion between academics and industrialists, with science as a new cement in place of religion (Morrell and Thackray 1981).

1.1 How Babbage Was Concerned with the Principle of the Division of Labor

Babbage's treatise *On the Economy of Machinery and Manufactures* (1832) resulted from a thorough investigation carried out since 1819, visiting the main industrial towns as far as Scotland,¹ where he met the philosopher and mathematician Dugald Stewart (1753–1828), who wrote about the division of labor in his *Lectures on Political Economy* (1835). With the *Philosophy of Manufactures* by Andrew Ure (1778–1857), also published in 1835, Babbage's treatise still stands as a main source, analyzing the nascent industrial world. It was abundantly quoted by Karl Marx (1818–1883) in Book II of *Das Kapital* (1867–1894). Babbage's Chapter 19 was devoted to the division of labor, and Chapter 20 to the division of mental labor.

In fact, the interest in applying the principle of division of labor in science had already been formulated in 1821, in the inaugural address of the Royal Astronomical Society, a new common center of communication and classification that Babbage contributed to the creation of with a group of "business astronomers" (Ashworth 1994). This address asserted that this principle would enable the efficient establishment of a complete catalog of stars and other celestial bodies from a minute and systematic examination of the heavens, planned by the theoretical astronomers. In this way, local material could be rationally collected from observers scattered all over the world, which would avoid the "loss of labour which must result from several [of them] working at once on the same region, while others are left unexamined" (Anonymous 1821, 1–7).

In his treatise *On the Economy*, Babbage developed the advantages and effects of the division of labor that he considered insufficiently clarified by previous authors. This principle was first conceived for saving time and money by reducing each production to a series of limited and simple operations. "This improvement" was first conceived for manual work, but was "generally the first step towards a machine" (Babbage 1832, § 224). The efficiency of organizing work in firms according to this principle was then inventoried and analysed as the best way to save time and money, improving the dexterity of workmen who repeated the same operation continuously. The operative view was systematically put forward throughout the book, the word "operation"

1 When he met Dugald Stewart, Babbage discussed with him the second volume of the latter's *Elements of the Philosophy of the Human Mind* (1792–1827), which had just been published in 1814. Stewart also popularized the works of Adam Smith.

being used for both industry and mathematics, and meaning any kind of transformation:

[The] effect of the division of labour, both in mechanical and in mental operations, is that it enables us to purchase and apply to each process precisely that quantity of skill and knowledge which is required for it: we avoid employing any part of the time of a man who can get eight or ten shillings a day by his skill in tempering needles... and we equally avoid the loss arising from the employment of an accomplished mathematician in performing the lowest processes of arithmetic. (Babbage 1832, 141)

In fact, this efficiency clearly presupposed a hierarchical organization, inducing a concentration of power in the hands of the organizers themselves, who determined how the division of labor would be carried out. Better efficiency would be attained if everyone worked at his right pace, workers and capitalists—a very common word in Babbage's vocabulary—for the economy, and calculators and theoretical mathematicians for science.

The applied sciences derive their facts from experiment; but the reasonings on which their chief utility depends are the province of what is called abstract science... the division of labour is no less applicable to mental productions than to those in which material bodies are concerned; and it follows, that the efforts for the improvement of its manufactures which any country can make with the greatest probability of success, must arise from the combined exertions of all those most skilled in the theory, as well as in the practice of the arts; each labouring in that department for which his natural capacity and acquired habits have rendered him most fit. (Babbage 1832, 261)

Babbage looked at industry from the capitalist perspective. He never considered the workers' loss of "know-how" and understanding of their jobs. But he worried about the turbulent development of the industrial world. For him, the main reason for so much trouble in industrial towns was the lack of coherent development of factories, which had to be regulated by government, and guided by scientists in a Baconian manner. In the present state of affairs, as a Radical, Babbage violently criticized the lack of scientific knowledge of the governing class, which he considered a serious shortcoming:

It is in some measure to be attributed to the defects of our system of education, that scientific knowledge scarcely exists among the higher classes of society. The discussions in the Houses of Lords or of Commons, which arise on the occurrence of any subject connected with science, sufficiently prove the fact. (Babbage 1830, 4)

The maxim that "knowledge is power" can be perfectly comprehended by those only who are themselves well versed in science (Babbage 1830, 16).

In order to maintain the threatened social harmony, and their own power, English statesmen had first to understand what was happening in factories, but even more essentially, to coordinate industrial development and to support the work of scientists, mainly mathematicians, who were considered since Isaac Newton (1642–1727) as elaborating the mathematical principles of natural philosophy:

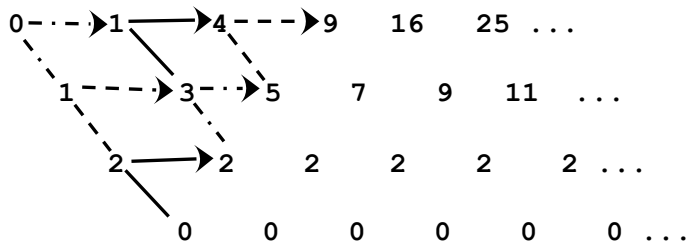
[The] discovery of the great principles of nature demands a mind almost exclusively devoted to such investigations; and these, in the present state of science, frequently require costly apparatus, and exact an expense of time quasi incompatible with professional avocations. It becomes, therefore, a fit subject for consideration, whether it would not be politic in the State to compensate for some of those privations, to which the cultivators of the higher departments of science are exposed; and the best mode of effecting this compensation, is a question which interests both the philosopher and the statesman. (Babbage 1832, 261–2)

Babbage was here explicitly expecting help from the government to build his engines. The division of labor he wished to establish in mechanizing computations was intended to free natural philosophers from tedious tasks in order that they could devote their time and thought to discovering the deep meaning of the organization of nature, and help the governing class to manage the organization of the industrial world according to natural scientific laws.

1.2 How the Difference Engine Exemplified This Principle

Babbage's reflections on mechanizing mental work directly issued from his work on his Difference Engine, which he designed and began to build in parallel with his enquiry on manufactures, also since 1819. Here too, the organization of work according to the principle of the division of labor preceded its mechanization. Babbage took as a model the way the French engineer Gaspard de Prony (1755–1839), then director of the *Ecole des Ponts et Chaussées*, organized manual computations for new tables of logarithmic and trigonometrical functions. This work became necessary as the French Revolution decided to introduce the metric system, including angles measured in decimal “grads” (Babbage 1832, 192). As de Prony was first discouraged by the enormity of the task, he was relieved to discover how Adam Smith's principle of the division of labor could help him organize the computations.

He used the method of differences, by which the values of a function at successive points could be evaluated by a series of additions, knowing only the initial values of the function at a given point and of its differences. Figure 1 is an example for the function $f(x) = x^2$, where the second differences are constant and the third differences are zero. Arrows in different styles in Figure 1 show how various additions are carried out to obtain the successive values of



[Fig. 1] Example of the method of differences for $f(x) = x^2$

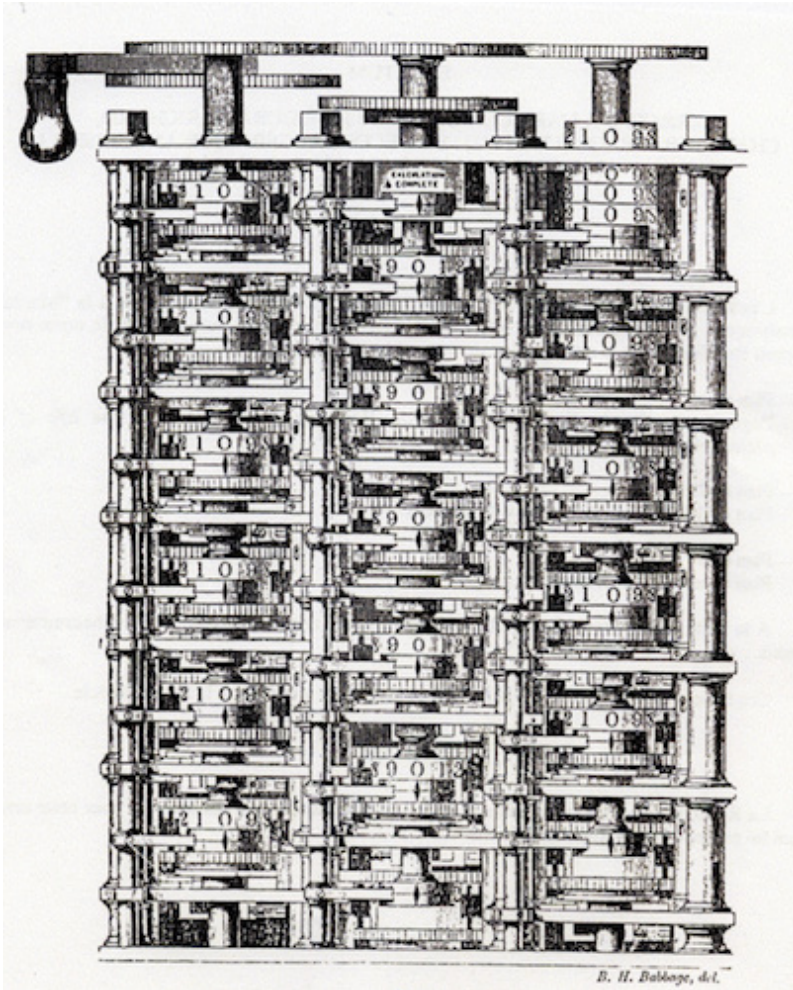
the differences and of the function, starting from the initial values on the left,² and from the null difference on the last line.

This method gave exact values of a polynomial function—the last constant difference corresponding to its order—and a good approximation of any continuous function on intervals previously chosen.³ For instance, the null fourth difference could give an approximation of order 10^{-10} for the logarithmic function, from $x = 10,000$, with a step $h = 1$ (Lacroix 1819, 12–14).

In de Prony's organization of work, a small group of five or six eminent French mathematicians—the first section—chose the more interesting expansion in series of the function whose values had to be evaluated. A slightly larger group—the second section—decided the intervals of the function and the step of the variable. A third section, more abundant, executed only additions or subtractions, working almost mechanically without thinking about the meaning of their work (Babbage 1832, 135–38; Grattan-Guinness 2003). Babbage's Difference Engine mechanized the work of the third section. He underlined that “from that part executed by the third class, which may almost be termed mechanical, requiring the least knowledge and by far the greatest exertions, the first class were entirely exempt” (Babbage 1832, 137).

The number of columns on the machine gave the number of differences it was intended to use, with one more from the values of the function. The number of wheels on each column gave the number of digits of each figure. The initial figures were placed on the wheels before starting the work for each difference and for the function. When an operator turned the crank at the top of the machine, a gear mechanism added the figures from a column to the following one, up to the function.

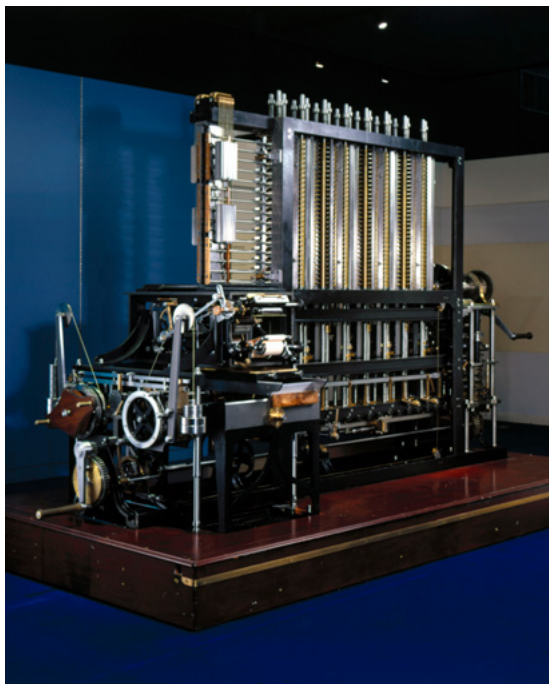
2 $0 + 2 = 2$, $2 + 1 = 3$, $1 + 0 = 1$, and so on.
 3 This property was used with confidence by the mathematicians of this time. It will be fully demonstrated later by the German mathematician Karl Weierstrass (1815–1897). The Weierstrass approximation theorem established that any continuous function defined on a close interval can be uniformly approximated as closely as desired by a polynomial function.



[Fig. 2] Assembled part of Babbage's Difference Engine n° 1. Frontispiece of *Passages from the Life of a Philosopher*, 1864.

Babbage's approach was not a simple analogy between mechanical and mental operations from Adam Smith's principle, which he described however in detail (Babbage 1832, 138–39). It was mostly an attempt to found the former on the latter, so as to show the natural character of the new form of work in factories. He wrote:

It will afford an occasion for shewing that the arrangements which ought to regulate the interior economy of a manufactory, are founded on principles of deeper root than may have been supposed, and are capable



[Fig. 3] Babbage's Difference Engine n° 2, realized for the Science Museum (1991–2000) from plans 1847–49. Science Museum/SSPL. London. Inv. n° 1992-0556. Plate n° 10324907.

of being usefully employed in preparing the road to some of the sublimest investigations of the human mind. (Babbage 1832, 135)

Babbage was all the more sensitive to the operative approach of labor and to the mechanization of its process, in factories as well as in mathematics, as it fit in well with the symbolical approach that he was developing in the same years at Cambridge with his algebraist friends. This approach essentially emerged from an extensive correspondence and joint initiatives with George Peacock (1791–1858) and John F. W. Herschel (1792–1871), questioning inventive and generalizing processes in algebra. It was fully exhibited by Peacock from 1830. Symbolical algebra focused on operations that were no longer defined by their operating technique or by the meaning of their results, but by the laws of combination which ruled them as transformation processes. Introducing a strict separation between operations and data, it claimed the independence and universality of the former from the contingency of the latter. In such a way, the demonstrative process followed step by step the unfolding of the transformations governed by these combination laws alone, without

necessary reference to the data or to the numerical values of the results. So, algebra became a calculus expressing the operations of the mind in a Lockean approach of understanding, dealing with general meaningless symbols to which a meaning could possibly be attributed—or not—afterwards (Durand-Richard 2020; Peacock 1830). The Difference Engine somehow embodied such a strict separation in the special case of the method of differences, and the Analytical Engine will proceed in the same way for the whole mathematical analysis after 1834.

1.3 The Failure of Babbage's Enterprise on His Engines

Although Babbage set up the separation between operations and data—which will be fundamental in future computers—and although he is often considered as a “pioneer of the computer” (Hyman 1983), he did not succeed in realizing his engines. He built a first model of the Difference Engine in 1822, with two orders of differences. The plans he produced in 1830 for the Difference Engine No. 1 included six orders of differences, 20 decimal places, and a printing mechanism, so as to prevent copying errors, but they were only partially realized (Babbage 1832, 139),⁴ and Babbage stopped the work in 1833. His attention then shifted to the design of the Analytical Engine, for which many plans were elaborated and some elements built, but which was never completed.

Many reasons have been put forward to explain this failure, which could be resumed in saying that the social and technical environment was not entirely compatible with the achievement of such projects (Dijkstra 1975, 546). Undoubtedly, Babbage himself gave the main reason which could explain this failure, when he wrote that “The division of labour cannot be successfully practised unless there exists a great demand for its produce; and it requires a large capital to be employed in those arts in which it is used” (Babbage 1832, 201).

Historical analysis focused on financial and practical difficulties (Hyman 1983, Swade 1996). But the main issue was more profoundly the discrepancy between the ambition of Babbage's project, its reception by his contemporaries, and his own working practices. Babbage's goal, especially with the Analytical Engine, was to mechanize computations for the whole mathematical analysis. But for his possible funders, the specific needs for such computations were too far removed from their concerns to induce their investments. Even though the government allocated up to £17,000 to support the first project for 10 years, and Babbage devoted his own personal fortune to his researches on the Analytical Engine (Dubbey 1978; Hyman 1983), these

4 A large model was set inside the Science Museum in London, and is now exhibited, (cf. Figure 2) from new plans established between 1847 and 1849 (Swade 1996).

efforts were wretched compared with the huge amounts spent during the Second World War, as it will be shown below, that led to high-speed electronic machines. Moreover, in the Victorian period from 1837, social interest in experimental sciences gave precedence to materializing the operations of the mind: in the following years, the installation of transoceanic telegraph cables by William Thomson (1827–1907)—the future Lord Kelvin—and his associated engineers will receive more attention than the construction of the Analytical Engine, as praised by one of Babbage's sons (Babbage 1910).

Moreover, paradoxically, Babbage did not develop relations with the industrial world at all in his working practices, whereas he always defended their importance for a governing class guided by scientists. In spite of his impressive enquiry on factories, Babbage persevered working as a “gentleman amateur,” with skillful craftsmen working in an impressive workshop set up at home especially for this purpose, and checking by himself each step of the making. Of course, part of this checking was certainly necessary at the beginning, as some pieces were just invented before being produced. If Babbage asserted that “the arts and manufactures ... are intimately connected with the progress of the severer sciences” (Babbage 1832, 379), he only contemplated this connection between people of equivalent social rank: statesmen, manufacturers, scholars. The British Association for the Advancement of Science, which he contributed to the creation of in 1831, was a good example of the mode of relationships he wanted to establish, where the liberal Anglican theology remained the implicit creed⁵ (Babbage 1832, 263; Morrell and Thackray 1981, 225–27). Regarding the work contents, whether manual or mental, the division of labor imposed a top-down hierarchy, and Babbage never considered bottom-up connections. He offered advice to workers not to suffer too harshly the rigors of capitalist modes of production, and always considered human calculators as technical workers, mechanically carrying out elementary operations, and able to make errors, so that it would be better to replace them with an operative mechanism. With this hierarchical position, Babbage never took care of the deskilling of workers, or calculators, induced by the division of labor in firms or computations.

The issue raised now is to examine how the conditions of production evolved so that mathematical instruments and machines would be effectively built and socially included in scientific work. As was just underlined, these devices were concrete objects, and so did not concern only the inventive spirit of scientists—often the most emphasized—but also their material production in a political and economic context had to be considered (Barbin 2020, 104). New

5 “If the British Association managers sedulously cultivated a non sectarian, inclusive, and non political public image for their science, the truth was far otherwise. No Jews, Romans, Catholics, Methodists, Congregationnals, Baptists, declared Tories, Utilitarians, or proponents of socialism & of the working classes were to be found in their members” (Morrell and Thackray 1981, 26).

relationships had to be established between science and industry—not only those Babbage considered—which prepared and accompanied their design and achievement. While they were not given any attention by Babbage, they have now to be analyzed so as not to miss their consequences for the situation of scientists, operators and users, and on the impact of these machines in society as well.

2. Detours of Computations by Analog Machines

Initial works on the history of computing tended to consider that nothing happened between Babbage and Turing's paper on computable numbers (Turing 1936–37), erecting it thus, in an arguable and contested way, at the origin of modern computers. This trend of thought illustrates not only the fact that this history was initially written in a retrospective way, with views only on mechanizing digital computations, but also that these narratives were first written by actors, academics, and computer scientists impressed by these new high-speed machines often described as electronic brains (Berkeley 1949; McCulloch 1949; Segal 2003, 162–75). Observing mainly the formal and mathematical aspects of the development of computers led them to consider Babbage as ahead of his time, forgetting that ideas are not supported and transmitted only on their own, but that they need special attention in a specific context and institutions to support their recognition. The specific needs that were missing at the time of Babbage's analysis appeared during the 19th and 20th centuries, largely spurred by applied mathematics and experimental physics, and often in a non-academic context. These new needs were first satisfied by analog instruments and machines, produced by engineers and physicists rather than by pure mathematicians, and whose running often involved them together more collectively.

2.1 Analog Devices in Applied Mathematics and Experimental Physics

Since Babbage's works, considerable changes happened regarding the interest in mechanizing mathematical processes, in territories often outside universities. Numerical needs appeared more clearly and with more profit than at Babbage's time since natural phenomena and experimental physics began to be studied in close relation with the establishment of state infrastructures and industry. New computational needs came mostly from engineers and physicists. A lot of methods and analog instruments were produced "en masse" from the beginning of the 19th century, especially for integrating functions and supplying numerical solutions for differential equations in very wide fields of applications, when formal solutions—analytical ones—were impossible to find as mathematically non-existent or impracticable (Tournès

2022). Consequently, mechanizing mathematical computations turned out to be necessary as well as useful and profitable.

Analog devices like planimeters, integragraphs, and integrators gave integrals of functions whose analytical forms were unknown, such as the outline of a forest on a map, or the recorded curve evaluating the work-function of horses or machines in industry. Harmonic analyzers decomposed tidal curves recorded on a tide gauge, and gave the coefficients of the expansion in Fourier series of the corresponding function. Coupled with the tide predictor, the harmonic analyzer was crucial first in studying tides, before accompanying the development of harmonic analysis in experimental sciences all along the 20th century until the 1960s (Durand-Richard 2010, 2016). Finally, there were differential analyzers of the kind developed by Vannevar Bush (1890–1974) and his team at the Massachusetts Institute of Technology (MIT) in 1927 (Bush 1931). These machines largely supported the resolution of differential equations in various domains of physics. These devices largely contributed to the restructuring of spaces in industrializing countries, where engineers played a crucial role as a new socio-professional milieu, with institutions and journals, and a mathematical background acquired either directly in firms, or in polytechnic institutes and schools created all through the 19th century (Durand-Richard 2022).

The differential analyzer marked the transition from analog instruments to analog machines as “each process, by which any article is produced [is reduced to] a very limited and simple operation” (Babbage, 1832, § 224).⁶ Some of its working methods and fields of application will be largely transferred to digital computers, such as the Southwell method of relaxation and wind tunnel data processing (Hartree 1949a, § 3.4), diagrams used to set up the machine for each problem or series of problems, and libraries of partial settings, later named subroutines (Durand-Richard 2018). Garrett Birkhoff (1911–1996) insisted that for engineers and scientists, the differential analyzer and International Business Machines’ (IBM) electromechanical Automatic Sequence Controlled Calculator (ASCC) Mark I (1937–44–52)⁷ by Howard H. Aiken (1900–1973) kept the same potentialities and were still used at least up to 1950 (Birkhoff 1980). During the war, three US centers were equipped with these analyzers: The Moore School of Electrical Engineering, the Ballistics Research Laboratory at Aberdeen Proving Ground, and the MIT (Bullynck 2018). In England, the two differential analyzers,⁸ whose construction was initiated by the physicist and mathematician Douglas R. Hartree (1896–1958)

6 More than a century later, Hartree will give a classification where he distinguished a machine from an instrument, the former being completely automatized and not requiring any human intervention during its running (Hartree 1949b).

7 The three dates given for machines respectively indicate the start of their design, their achievement, and the end of their use.

8 They followed the plans of the one built by Bush (Hartree 1938).

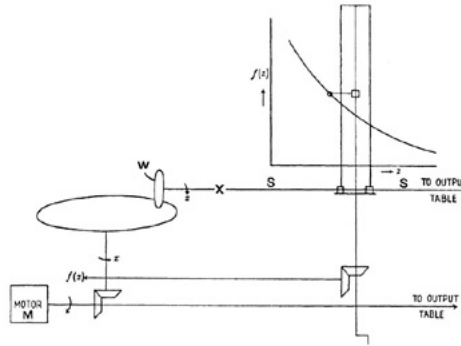
at Manchester and Cambridge universities just before the war, preceded the design and construction of the EDSAC (1946-49-78) by Maurice Wilkes' (1913-2010) team at Cambridge. At the end of the war, the Whirlwind (1944-51-83), whose development was at the core of the US Air Force defense system SAGE (Semi-Automatic Ground Environment), was initially conceived as an analog device, like other flight trainers it was intended to generalize.

The experience of Hartree in this matter is all the more interesting as he was already an expert in hand-made numerical computations. He contributed much to the transition between analog and digital devices, and published many papers on the two. Originally designed for scientific work in various departments, they were involved in physical applications to the national war effort for its entire duration: external and internal ballistics, automatic following of targets, heat conduction, and wave propagation by radio, radar, magnetron, and blasts.

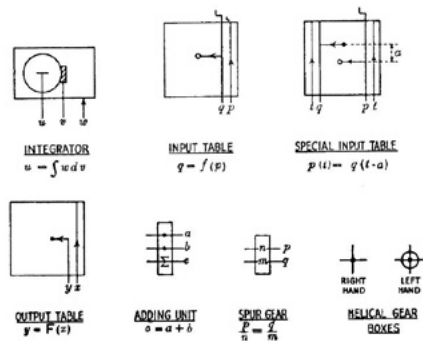
2.2 The Conceptual and Social Organization of Work on the Differential Analyzer

Analog instruments such as planimeters did not materialize any division of mental labor in their operating mode, since their mechanism directly gave the integral of a function by a continuous motion. A physical phenomenon was measured as a function of time recorded on a sheet of paper, and this graphical representation was read by an operator following the curve with a pointer. This displacement was transmitted inside the instrument to a mechanism, the integrator, which decomposed it in two orthogonal motions. On the disc and wheel planimeter for instance, the horizontal disc turned around its axis and drove the wheel which touched it at the distance $y = f(x)$ from its center. So, what corresponded to an infinitesimal displacement dx of the pointer gave $d\theta = f(x).dx$ as the angle of rotation of the wheel. The mechanism of the device recorded the total angle of rotation, which is the sum of $f(x).dx$, the sought integral (Figure 4a).

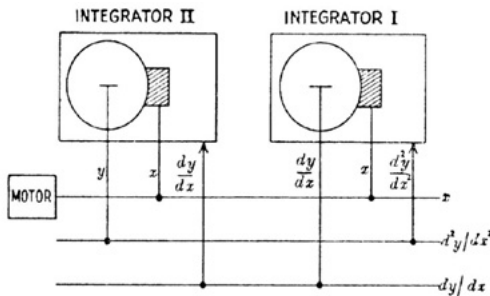
In fact, the division of mental labor appeared in the transition from analog instruments to analog machines, with the differential analyzer. Thanks to the torque amplifier—using the principle of a capstan—Bush succeeded in associating two integrators in such a way that the result of one of them could be reintroduced as an input in the second. So, he made possible the mechanical integration of linear differential equations of the second order by successive approximations, including those with variable coefficients (Durand-Richard 2010). The separation between operations and data inaugurated by Babbage's engines was maintained. Initial conditions were still introduced in the starting machine setting, and the coefficients of the equation fixed the scales between the various parts of the machine and their motions. But a new



[Fig. 4a] The mechanism of an integrator (source: Hartree 1938, 344).



[Fig. 4b] Notations for units of machine on set-up diagram (source: Hartree 1938, 351). “The letters $p, q, r, \dots$ stand for the quantities whose values are measured by the rotations of the respective shafts. Under the symbol for each unit is given the formula showing the relation between these rotations”.



[Fig. 4c] Set-up of the differential analyzer for the equation $d^2y/dx^2 = -y$. Scales and signs disregarded (source: Hartree 1938, 352)

category of data appeared, the curves of functions—more often empirical ones—presented on tables and followed by an operator for the input.⁹ This distinction could not be apprehended by Babbage, as the differential equations now numerically solved were much more complex than the cases he considered—either the simple evaluation of function values on the Difference Engine, or that of an algebraical formula mechanically translated on the Analytical Engine. This new type of data will also occur as numerical tables in digital computations (see section 3), the manipulation of which will give rise to new problems for organizing the work.

Elementary operations—including integration as a specific one—were mechanized and represented as units (Figure 4b). Their relations, given by the equation, were materialized by mechanical connections between these units. The process of working out a machine set-up materialized the composition of a mathematical formula from its elementary operations (Figure 4c). A preliminary analysis of a submitted problem had to express the physical phenomenon by a mathematical equation, and to write it in a form transferable on the machine. Then Hartree distinguished three main stages:

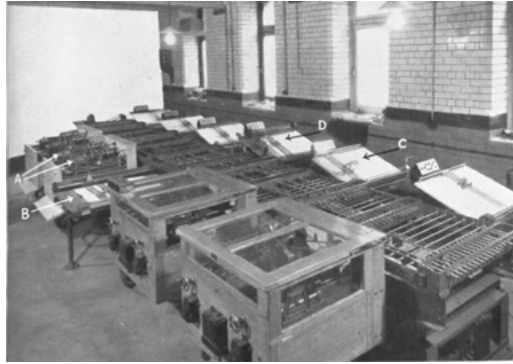
1. A diagram had to be established with a specific notation¹⁰ for each unit. It indicated how to connect these units.
2. Scales had to be determined for the rotations of the various shafts, so as to fix the ratios for gear trains between them.
3. The machine was set up according to the diagram. Hartree considered this set-up as the mechanical translation of the equation.

These three steps could last several days, but when the machine was set up, a solution could be obtained in 15 minutes (Hartree 1940). So the machine was more useful when a large number of solutions of the same equation was required—for instance with different initial conditions, or with different coefficients, as the form of the equation remained the same and the set-up of the machine did not change, only its starting position as well as some scales. This machine was first designed to evaluate ordinary differential equations, but Hartree adapted the method to deal with partial differential equations (Durand-Richard 2018).

Unlike in Babbage's time, scientists were now specialized by disciplines: mathematicians, physicists and engineers replaced the natural philosopher. And the design and setting up of differential analyzers required a collective approach, even with the industrialists who submitted the problems in their physical form, unless it was protected by secrecy. Hartree particularly emphasized it in his monograph handed in to the Headquarters Section of the Department of Scientific Research of the Ministry of Supply in 1949, reporting

9 Those obtained as output by the running of the machine could also be presented in tables.

10 This notation was introduced by Bush.



[Fig. 5] General view of the differential analyzer at Manchester University. Integrators coupled by pairs thanks to the torque amplifier are in the wooden boxes on the left side of the machine. Photo Metropolitan-Vickers Electrical Co. Ltd (source: Hartree 1938, plate II).

in great detail the methods used on the two Manchester and Cambridge machines during the war (Hartree 1949a). As the differential analyzer was not a wholly automatic machine, Hartree noticed that its setting and use required judgment, and an experienced group able to examine the submitted problems and to discuss “programs” and results with those who brought the problems. He asserted that in his group, the work was so collective that it was impossible to ascribe definite parts of the work to definite individuals. The machine and the research team should be considered as a unit:

It is not good to have a mathematician in one place preparing the problem for the machine, and operators, unskilled at anything but routine running and maintenance, actually handling the machine. ... Preferably most of the group should have a full understanding of what the machine is doing, and ... ready to undertake any of the auxiliary work which may be relevant to the machine treatment of the problems. (Hartree 1949a, 1.0.2)

Anyway, Hartree always insisted on judgment as reserved to humans. When he published his comparison between analog and digital high-speed machines, where he insisted on the entire automaticity of the second, he endeavored to separate the vocabulary related to the human and that related to the machine. He refused the term “judgment” to qualify the possible conditional jump in a digital high-speed machine: this was totally determined beforehand, while “judgment” implied free will and power of decision. For Hartree, who kept in mind the computational organization of Manchester University and the Cambridge Mathematical Laboratory, computation meaning was the job of the researchers working on the machine. A prime condition of scientific work was to establish a precise terminology for these new machines, which took care of not mixing up human and mechanical potentialities.

3. From Manual Processes to Punched Card Machines in Scientific Computations

Alongside analog instruments and machines for engineers and physicists, other fields of knowledge also required numerical results, and their needs considerably increased over the 19th and 20th centuries, especially in astronomy, ballistics, statistics, and business affairs. In these fields, as in the case of De Prony to Babbage, the division of labor was first gradually introduced into computations by hand, before partial mechanization occurred. At this point, public and scientific concerns for computations began to find some common views on how to proceed.

3.1 The Paradigmatic Reluctance of Astronomy

Astronomy represented a significant case of this evolution, as it dealt greatly with approximation methods, and concerned not only astronomers, mathematicians and human computers, but also a lot of users such as seamen, naval personnel, military officers, explorers, and boatmen. It should not be forgotten that Babbage designed his Difference Engine to produce numerical tables required for astronomy, when he created the Astronomical Society¹¹ with Herschel and their friends in 1820, in a period when the scientific expertise of both the Royal Society and the Greenwich Observatory was decreasing (Babbage 1830; Norberg 2003, 193). In fact, Babbage reacted essentially as a mathematician. But the renewal prompted by Pierre Simon Laplace's *Traité de Mécanique Céleste* (1798–1825) on the theoretical level and by the improvement of observations and Friedrich Bessel's methods of their reduction revealed that handling astronomical computations was a much more complex activity. George B. Airy (1801–92), head of the Cambridge Observatory as Plumian Professor of astronomy and experimental philosophy (1828–1835), and then Royal Astronomer at Greenwich Observatory (1836–1881), had a more complete view. He was convinced neither by the symbolical approach of algebra nor by the mechanization of astronomical computations. Consulted as an expert by the government in 1842 on the interest of continuing Babbage's project, he gave a completely negative response, as he considered it did not answer to the whole needs of the observatory:

An absurd notion has been spread, that the machine was intended for all calculations of every kind. This is quite wrong. The machine is intended solely for calculations which can be made by addition and subtraction in a particular way. This excludes all ordinary calculation. Scarcely a figure of the Nautical Almanac could be computed by it. Not a single figure of the

11 This society obtained its Royal Charter in 1831, and became the still existing Royal Astronomical Society.

Greenwich Observations or the great lunar Computations now going on could be computed by it. Indeed it was prepared only for the computation of new Tables (as Tables of Logarithms and the like), and even for these the difficult part must be done by human computers... I can therefore state without the least hesitation that I believe the machine to be useless, and that the sooner it is abandoned, the better it will be for all parties. (Airy 1842)

Airy was nevertheless concerned by the division of labor. He organized astronomical computations at Greenwich Observatory in a very hierarchical way which brought him great fame. Each step of this hierarchy dealt with a specific level of mathematical qualified knowledge, and each human computer could climb some rungs once he acquired the related knowledge. Observers had to carry out their own reductions. Calculators had to perform correctly elementary arithmetical operations, including fractions and logarithms. Several grades of assistants had to master different levels of mathematical knowledge, from rules of algebra to differential and integral calculus, and optics and analytical mechanics with reference to gravitational astronomy. And the algorithmic part "remained the Royal Astronomer's responsibility," helped by his assistants. Moreover, elementary arithmetical computations were presented on printed forms where they were decomposed into repetitive tasks, carried out by unskilled computers (Aubin 2009, 277–83). The use of tables also required great attention, for instance to extract results from several tens of tables to obtain an ephemeris "containing heliocentric and geocentric position coordinates of [a] planet" (Norberg 2003, 190), a task which was then difficult to conceive as mechanizable. At this point, even if some steps of astronomical computations were organized by the division of labor, their meaning had to be followed from one step to the other, so as to handle the continuity of the corresponding operations.

What Babbage expressed as a wish to promote the mechanization of computing began to be realized only with the abundance of needs and the availability of accounting machines with new performances. These simple mechanical calculators had started being developed to cater to the growing needs of business enterprises like manufacturers, insurance companies, and banks. These and other office machines were developed especially in the USA from the late 19th century onward and their manufacturers would later play a key role in computer history, most notably IBM. In the early 20th century accounting machines started being used for scientific aims to help satisfy increasing computational needs, as was the case in astronomy.

Publications of astronomical data were not very numerous when Babbage wrote before the 1830s. Airy began to deal with more material, applying his hierarchical organization to handle observations lying at Greenwich since 1750, so as to improve the orbital elements of planetary theory (Norberg 2003,

193). As the number and precision of observations increased, and as celestial mechanics refined its theoretical formulations, discrepancies appeared between observation and theory, which required more precise tables. And the relationships between theory, computations and observations fluctuated throughout the century. The production of results in astronomy and mathematical tables grew considerably in number and accuracy, but manual computations remained the main practice of astronomers, and required very long processes, even if desk machines could be occasionally used. For instance, the three volumes of *Tables of the Motion of the Moon* (1919) by Ernest W. Brown (1866–1938) resumed 12 years of work and contained 180 tables with explanations of their use (Olley 2018, 4–5). They were based on his new “Theory for the Motion of the Moon,” published in five parts in the *Memoirs of the Royal Astronomical Society* (1897–1908), including new planetary constants, better accuracy, and effects of more planetary perturbation. They were realized by Henry Hedrick, helped by four manual computers (Brown 1919, vii). Likewise, the *Nouvelles Tables Trigonométriques Fondamentales* (1915–1918) by Henri Andoyer (1862–1929) gave the values of these functions to 15 decimals at intervals of 10′. This author noted that he carried out the computations entirely by himself, and that these tables with “natural values” were better suited to the ever-increasing practice of calculations, executed directly using machines without the help of logarithms (Andoyer 1915, v). But this use of machines in astronomical computations was still not very well considered at the beginning of the 20th century (Comrie 1925, 243). From the printed sheets of formulae and precepts that remained as rare traces of usual practices, the methods for computing astronomical and nautical tables took a long time to evolve at Greenwich and the Nautical Almanac Office (NAO) (Wilkins 2003, 300). When Donald H. Sadler (1908–1987) began his career as assistant at this office in 1930, before succeeding Leslie Comrie (1893–1950) in 1937 as superintendent, he remembered the division of labor between skillful qualified computers such as the Daniels brothers, and the “routine” jobs of the junior staff, even if mechanization began to be introduced, first dealing with repetitive tasks (Sadler 2008, 19–21).

3.2 Mechanizing Computations in Astronomy

As soon as he became Deputy Superintendent of the NAO in 1925, Comrie launched a program for comparing machines available on the market. He wanted—and within six years succeeded—to convince colleagues and institutions to use them in astronomical and nautical computations, rather than to design and build very expensive special machines. He hired machines from commercial firms so as to demonstrate their low cost in time and money. This new use also imposed a radical change in computing practices, focusing on the method of differences as more “natural” than the usual logarithmic

method of Carl Gauss and Bessel, because it was “directly founded on the first principles of functions” (Comrie 1925, 244), as already asserted by Andoyer. Comrie largely referred to Babbage, and to the few very expensive difference engines built in the 19th century for specific contexts (Williams 2003). But he adopted a much more pragmatic and bottom-up approach, relying on existing resources rather than on a whole program for reforming the field. Accounting machines became cheaper, more reliable and more sophisticated, and in 1912, T. C. Hudson had already introduced a Burroughs Adding Machine with one register at the NAO to calculate the values of functions from their second differences, through numerous manual manipulations. In his footsteps, Comrie understood that the extra register—a storing or accumulating register—of the new Brunsviga Dupla machine, coupled with a mechanical transfer, offered “greater possibilities than any single existing machine” and a “general adaptability to the needs of scientific computers.” With these two registers, the method of differences and its double summation could be entirely mechanized, as well as the classical interpolation of functions by the method of differences and the way to obtain second differences directly from the values of the function—a useful case also for statistical tables. The Brunsviga Dupla could also perform the four arithmetical operations, and more easily thanks to new automatisms. Comrie detailed all the possible cases of its uses he found “simple”—compared with computations by hand—and “really fascinating.” He focused on a mathematical problem issued from least squares, correlation, and curve fitting work: to determine the products line by line of two columns of numbers, of their squares, as well as the sums of each given column and of the resulting ones. Examples were also given for astronomy, such as how to obtain and control the mean positions of stars for the annual precession, the mean longitudes, longitudes of nodes, perigees, and moments for finding means and standard deviations. Comrie underlined that these new possibilities were unsuspected by the builders and could inspire human computers since it was “now within the reach of all scientific institutions” (Comrie 1928, 459). The process could be extended with other machines, such as the Ellis machine with four registers, and the National machine with six registers, equivalent to Babbage’s project of his Great Difference Engine.

A next major step in mechanizing astronomical calculations occurred with the introduction of the punched card Hollerith machine, built by IBM. Annual ephemerides had to be composed from Brown’s *Tables of the Motions of the Moon*, which meant the summation of more than 1,400 periodic terms, with coefficients as large as 0°0003. In manual tabulation, data had to be taken in rows and columns of tables in an irregular way, depending on the specific orbital element to be obtained, before executing the appropriate summations. So, instructions were very complex, and the work very painstaking and a source of errors, requiring skilled calculators. So, if “copying was done once only, [if] the addition was done mechanically, and the results of the addition

printed," the task was already substantially reduced in terms of time, arduousness, skill, and cost. Comrie explained with details how the mechanical tabulation was modeled on the tabulation by hand, especially for calculating the long and short period terms of nutation (Comrie 1932). With the Hollerith machine, each table was represented by a pack of punched cards with 45 columns. Two new main technical elements for mechanizing the choice of rows and columns in the tables were the sorter—so as to separate punched cards into groups—and the tabulator—a high-speed adding machine from holes in the punched cards. But the initial mixing of these cards, corresponding to the appropriate choice of tables previously carried out by operators to calculate the ephemerides, was still made by hand (Sadler 2008, 22). When Wallace J. Eckert (1902–1971), a former student of Brown, founded the Thomas J. Watson Astronomical Computing Bureau at Columbia University in 1937, he inherited both Comrie's experience and machines working at the Statistic Bureau of his university. Flexibility of the machine was improved by ensuring the automatic control by a plugboard, analog to a telephone switchboard, by which the sequence of operations of the tabulator was controlled automatically (Eckert 1940, 12–14). His book *Punched Cards Method and Scientific Computation* (1940) much influenced early computer scientists.

But even if some tasks were automatically dealt with, human computers did not disappear, although their tasks changed. The work of the junior staff—often young women—was still repetitive, but changed in nature when routines were ensured by the machines (Eckert 1940, 52). In such agency, women operators had now to serve the machine, losing sight of the astronomical meaning of their task. Sadler underlined this aspect in a very significant way:

The junior staff did 'routine' work: mainly operating the Burroughs and, later, the National machines; tearing and pasting the printed results to make printers' copy; and systematic simple arithmetical jobs, with or without a calculating machine. Most systematic computations were planned to be done on numbered forms, with one date (or other argument) to the column. (Sadler 2008, 21)

Mrs Frayne was a most efficient punched-card operator and Miss L. H. Burr, among others, was a fast and accurate Burroughs operator. Miss Burr later operated the National machine... The methods devised and, rather more importantly, implemented in the face of great difficulties by Comrie revolutionised systematic computing. They required the willing cooperation of staff to do repetitive tasks at high speeds with a very high standard of accuracy; and this he obtained. Using spare machine-time whenever possible the unit-cost of the work was incredibly small. Man- and, particularly, woman-power was extremely cheap, and Comrie, as a matter of principle, insisted on making optimum use of staff and machines. (Sadler 2008, 23)

Comrie focused on the efficiency and low cost of the work carried out by commercial machines and women. The speed of production of astronomical computations increased in considerable proportions, since Eckert indicated that the sorter ran at the rate of 20,000 cards per hour, “the tabulator [operated] at the rate of 4500 cards per hour while listing, and 9000 while tabulating, and the amount of errors was sorely reduced” (Eckert 1940, 12–16). However, machines were still considered as aids to computations, as previously tables were, and not aimed to handle completely the whole mastery of computing practices. Moreover, even if these machines still existed on the commercial market, they remained expensive enough so that the question arose of the necessity of their use. The first question Eckert asked in opening his book was: “Have I enough operations of this type or that to justify such powerful equipment?” (Eckert 1940, 1). Regardless, Eckert’s experience on IBM’s machines meant he felt they were efficient enough to be called upon during the Second World War to supplement existing resources (Rees 1987, 833).

3.3 Other Fields Mechanizing Digital Computations

Mechanization was far from occupying the entire field of scientific computing at the end of the 1930s, but the use of the Electric Punched Cards Method extended it from business and statistics to other domains, from mathematical tables to applied mathematics, with a parallel evolution from manual to automatic modes.

On the one hand, the division of mental labor took a more systematic turn in manual computing practices with the Mathematical Tables Project, created in 1935 by the Works Project Administration (WPA), one important element of Roosevelt’s New Deal in 1933 (Grier 2003). The work was organized by Arnold Lowan (1898–1962), an engineer well versed in mathematics, and Gertrude Blanch (1897–1996), a mathematician working then for a camera firm, with the help of a few competent computers. Then, as in De Prony’s time, computations were broken down into elementary operations, presented on printed worksheets. They were carried out by groups of unqualified people tasked with a single operation, helped with old desk machines and devoid of any initiative. During its almost 10 years of existence, the project was somewhat decried by conservatives as just to fight against unemployment, and rejected by some scientists who, like Eckert and Comrie, promoted mechanical means. But it produced 28 volumes of mathematical tables later published by Columbia University Press, with also smaller ones extracted from them. Such tables were mostly needed for special mathematical functions occurring increasingly often with the developments of physics in the 19th and 20th centuries. Some machines, Monroe calculators and the IBM tabulator, were progressively introduced in 1941 in connection with military projects. When Roosevelt dissolved

the WPA project in 1942, its members were distributed between the Navy Hydrographic Office and the Applied Mathematical Panel, the mathematical branch of the Office of Scientific Research and Development (OSRD).

On the other hand, the same pragmatism and a course analog to that followed in astronomy could be observed in the mechanization of ballistics computations. Handmade computations preceded and guided their mechanization. The reluctance was not the same as in astronomy, because the needs for more results appeared later, when new factors had to be included to predict the path of new projectiles, facing their increasing speed and range. During the First World War, facing these new factors, the astronomer Forest R. Moulton (1872–1952) had to improve mathematical techniques. He devised a method for numerical integration of ballistics equations. In the following years, artillery officers published manuals where this method was presented step by step, in order to reduce the work of human computers to a series of instructions they had to follow scrupulously: to choose values in numerical tables and to carry out elementary operations—additions and some subtractions—on them (Bullynck 2018). Tables were both those of empirical functions obtained by experiments at the proving ground—retardation or Gâvre function, altitude function—and some mathematical tables—for squares, logarithms, and some operations already performed according to the values of the variables. Computing sheets had to be filled by human computers following the required instructions. So, the same kind of practice could be recognized as in astronomy: once the theoretical framework was drawn, to establish tables meant to carry out a procedure given by a series of instructions, acting on different types of data. Three different types of data could still be recognized, which stood independently of the procedure:

- the numerical data corresponding to the initial conditions of the problem
- the coefficients of the equation to be solved
- and tables of functions, numerical ones this time, still new compared with Babbage's approach.

With such a device, even if initial conditions could differ slightly, or if tables of functions were changed, the procedure remained stable, in the same way as the setting up of the differential analyzer.

This practice of ballistics computations not only made them ready to be mechanized, but suggested the very structure of the hardware of the first relay-based digital computers. In Models II (1943), III (1944) and V (1946) built by George R. Stibitz (1904–1995), a specific programming unit controlled tables of functions, a device totally absent in von Neumann architecture. Moreover, the first ENIAC inherited both, and Maarten Bullynck showed it inherited the organization of the original ballistics computations, particularly related to what kind of data would have to be consulted quickly or slowly (Bullynck 2018, 245–328).

So, throughout the 19th century and the first half of the 20th century, in parallel with analog machines, the division of mental labor was organized in more details than on the differential analyzer, so as to tackle complex numerical computations, requiring better approximations. Repetitive tasks—including operations—were carefully broken down in order to be carried out without any specific knowledge, so that it was possible to mechanize them. Moreover, the problems dealt with no longer originated solely from the scientific field—mathematics and natural laws—but also included direct social and economic concerns. By this very fact, mechanization was introduced in a pragmatic manner, initiated locally by individual actors, who had to convince institutions to find funding. They started relations with firms and adapted existing commercial machines to scientific needs.

Nevertheless, calculating machines were still conceived as aids to computations. Designing what Eckert named programs (Eckert 1940, 25) was a collective task mobilizing mathematicians, physicists, and engineers, as well as those who submitted the problems to be solved. So the meaning of computations was their fact: it needed to be negotiated between all actors, as it involved more than a strictly mathematical significance. It could also be followed and controlled on the machines, the set-up of which was visible. As for those who carried out repetitive tasks, the way they could grasp the significance of computations receded with the mechanization. Even if repetitive, their work had been mental. It became gestural when they just served machines.

4. A Turning Point: The Change in Scale of the Second World War

The First World War had already given rise to some specific relationships between science and defense (Aubin and Goldstein 2004), particularly with regard to mathematical computations, as evidenced by Hartree's first works in anti-aircraft gunnery (Darwin 1958, 105) or Moulton's method for numerical ballistics equations (see section 3.3). The Second World War gave a more fundamental impulse to the establishment of structural relationships between academics, industry, and defense. Firstly, new weapons required more precise firing tables accounting for hundreds of different sets of conditions, and they had to be produced quickly enough to be useful. Existing differential analyzers, in the United Kingdom as in the United States, were not enough to fulfill these new requirements. New projects were initiated for automatic computations, supported by a stricter organization associating more closely material and scientific means. While obtaining funding for calculating machines was always a daunting task during the interwar period, it became much easier with the outbreak of the war. What was previously left to the

initiative of individuals or small scientific groups became part of large government projects managed from the top at an industrial scale. The material conditions for realizing computational works changed radically, being included in a huge effort of scientific instrumentation.

[In] historical honesty, we have to realize that it was dedication to the struggle against Hitlerism, and later to other problems of national defence, that provided the main driving force behind the development of the computer in the 1940s. It's absolutely impossible to understand it except in that context. (Birkhoff 1980, 23)

In this new context, the division of labor became extreme across the entire organization of the war effort, so as to best concentrate on optimizing efficiency in every part of the work, the main goal of each enterprise remaining often secret locally and only known by the head organizers. Nevertheless, at the upper level, coordination required institutionally organized exchanges, to dispatch the various tasks at the beginning of the process and to synthesize the results so as to compare them with the initial demands—feedback which became more and more necessary when computations became increasingly complex. The following section focuses on what happened in the United States as the place where these coordinated efforts led to the development of a computer industry after the war.¹²

4.1 New Institutional Relations Structured by the War Effort

Since 1940 in the USA, the National Defense Research and Committee (NDRC), which became the already mentioned OSRD in 1941, brought together at least 6,000 scientists and built strong relationships between universities, industry in private laboratories—such as the Bell Laboratories or Western Electrics—and national defense. The MIT was the Navy's first partner at the end of the war. The OSRD was organized in divisions, each one directed by the president of a major university or research laboratory. Fire control and instruments were parts of the D division. For obvious security reasons, each team was unaware of the work of the others and only the section managers had an overview of the work carried out (Segal 2003, 87–93).

As its director, Bush himself noted in 1946 that the war catalyzed a new kind of interdisciplinarity, marked by effective cooperation at its highest levels between scientists, engineers, and the military. In 1942, he created the Applied Mathematical Panel (AMP), with Warren Weaver (1894–1978) at its head. Many of the mathematicians involved in computer design were part of it, such as

12 Space is missing here to examine the case of Bletchley Park in England, a real “factory of numbers” devoted to breaking German codes. The division of mental and material labor was very systematic there, organizing the work for more than 10,000 people by the end of the war, 75% of them women (Durand-Richard and Guillot 2019).

George Stibitz (1904–1995), Birkhoff and John von Neumann (1903–1957). The AMP was assigned to provide mathematical assistance to military services and other divisions of the OSRD, and set up contracts with 11 universities, including Columbia, Princeton, Berkeley, Brown, Harvard, North Western, and New York University. It inherited the Mathematical Tables Project (see section 3.3) and had to promote researches in applied mathematics—particularly in statistics, numerical analysis, and computing—and to coordinate them. As is well known, the construction of the ENIAC computer was funded by the Ballistic Research Laboratory at the Moore School of Electrical Engineering at the University of Pennsylvania, as even the Rockefeller differential analyzer was not efficient enough to meet the demand for firing tables (Rees 1987, 833).

4.2 Towards an Industry of Computing

At the end of the war, several naval officers associated with the OSRD, together with legislative and executive leaders of the government, were anxious to maintain the vitality of wartime research, and to establish channels through which scientists could continue to contribute to the nation's security in peace time. The Office of Naval Research (ONR) was created in 1946 by an Act of Congress, so as “to plan, foster, and encourage scientific research in recognition of its paramount importance as related to the maintenance of the future naval power, and the preservation of national security” (Rees 1987). It was mainly concerned with automatically sequenced electronic computing machines, and in the next decade the government would remain the main support for their production. Other civil and military institutions also strongly supported these developments, such as the National Bureau of Standards (NBS), the Atomic Energy Commission, the National Security Agency, and the ordnance offices of the army and navy (Rees 1987, 831). The main axes of its research program were still in pure and applied mathematics, statistics, computer development and numerical analysis, which became a fully recognized discipline in this process (Aspray 1989). Its staff constituted a small professional group with a high level of cooperation and commitment, which was going to play a central role in bringing new institutional structures to this research, and a lot of publications on computers acknowledged ONR funding. Members of the Mathematical branch of the ONR were involved in the creation of the Association for Computing Machinery in 1948 and contributed to its programs “to provide better communication among those interested in the new machinery” (Rees 1987, 831).

The ONR strongly contributed to the NBS, which also reinforced its mathematical research, creating four sections of the National Applied Mathematics Laboratories, namely the Computation Laboratory, the Statistical Engineering Laboratory, the Machines Development Laboratory, and the Institute for Numerical Analysis (INA) established at the University of California, Los

Angeles (UCLA). All these structures were coordinated by an Applied Mathematics Executive Council, which “provid[ed] an effective medium for the exchange of information among persons interested in the federal computing efforts” (Rees 1987, 837).

The design and construction of computers was supported by these structures and built by the NBS, academic institutions, or commercial firms. The machine built at the Institute of Advanced Studies (IAS) at Princeton University, whose construction was directed by von Neumann, and the Whirlwind project at the MIT were the two major machines of the ONR’s project. Among the machines built by the NBS itself were the SEAC (Standards Eastern Automatic Computer) and the SWAC (Standards Western Automatic Computer) at Los Angeles, both of them ordered in 1948 and completed in 1950. But a strong initial intention was also “that the machines ordered through NBS would be purchased from commercial sources in the hope of assisting in the establishment of an electronic computer industry,” the construction being monitored by the Engineering division of the NBS (Rees 1987, 837–41). The Census Bureau also ordered two machines: the RAYDAC (1947–1952) was built by the firm Raytheon, and the Electronic Control Company of John W. Mauchly (1907–1980) and J. Presper Eckert (1919–1995) built the BINAC (1949), but also three UNIVAC (1951–1952), of which 46 items were later manufactured. According to Mina Rees (1902–1997), head of the Mathematical branch of the ONR (1946–1953), “many small companies tried to enter the market, but it was not until the very large companies had made substantial financial commitments that the computer industry really took off” (Rees 1987, 834).

The INA has greatly contributed to making numerical analysis a discipline in its own right, providing optimal methods for the use of electronic computers, thus commanding respect from “pure” mathematicians at a time when they were still often reluctant to engage in what they considered only applications. The INA brought good mathematicians¹³ to UCLA, and trained new enthusiastic computer scientists who dispersed all through the USA. Also the ONR supported first courses on the subject, such as the one given at the Moore School of Engineering on the theory and techniques of electronic digital computers (1946), and Aiken’s graduate course at Harvard (1947–48). The first presentation of flow diagrams, by Herman Goldstine (1913–2004) and von Neumann in 1947, *Planning and Coding of Problems for an Electronic Computing Instrument*, was funded both by the Research and Development Service of the US Army Ordnance Department and the Institute for Advanced Studies. Some years later, the ONR sponsored with other institutions some conferences essential to the field, such as the first *Symposium on Automatic Programming* (1954) and the one on *Advanced Programming Methods for Digital Computers*

13 Its successive directors were prominent visiting professors: Hartree was one of them in 1948.

(1956). Both of them invited speakers from government agencies, universities, and industrial organizations.

Thus, the conditions of production of computations changed radically in the 1940s. The Second World War hugely reinforced a material organization that structured the landscape of scientific research for the future. It brought to the fore applied mathematics and sciences for defense and industry. More than a century before, Babbage wanted to appeal to the government not to let the capitalists organize industry, but rather to entrust scientists like him with the rational organization of the division of labor, whether material or mental. His approach was strictly top-down. From the Second World War onward, as scientists were strongly involved in societal developments, the issue was no longer to submit industrial developments to natural laws, but to express mathematically problems raised there. The development of computers resulted from these strong interactions between mathematicians, physicists, engineers, industrialists, and the military. In parallel, the separation was reinforced between design and execution tasks: if collective exchanges were favored between designers of mathematical problems and machines, the work was carried out in a process where elementary tasks were distributed among numerous operators, who at first executed mechanically elementary operations, and later acted as servants of the machines.

5. The Division of Mental Labor and Its Limits on the First Electronic High-Speed Digital Computers

The organization of computational research emerging from the Second World War therefore did not disappear when the war ended, and the Cold War supported the same kind of needs, as witnessed by the Whirlwind. In this context, mechanizing computations on the first electronic, digital computers could appear as the culmination of the division of labor, which at the same time was made invisible inside the machine. As electronic high-speed computers enclosed more and more whole processes previously visible to all, planning of computations had to be thought out in advance. And programming, the specialized activity so produced, turned out to be more difficult than expected with a simple binary coding of data and instructions.

5.1 The von Neumann Architecture on Computers

At the end of the war high-speed digital machines, beginning with ENIAC, radically renewed the computing field. The *First Draft of a Report on the EDVAC*,

which posterity attributes to von Neumann,¹⁴ synthesized the experience acquired on the ENIAC, and took one more step in presenting the “structure of a high speed automatic digital system” with the new idea of a stored program (von Neumann 1993 [1945], 25). Rather than its technical achievement, its description concerned the whole control of the computation processes, that is the correct sequencing of elementary operations according to the circuits of the machine. This so-called von Neumann architecture was not the only possible one, and some computers worked with parallel computations, such as the ENIAC, the BINAC, the SWAC, and the IAS. But its hardware was simpler, and its sequential approach was applied at all levels of its organization (Priestley 2011, 161). It would dominate for more than 30 years, and deeply mark the style of the first programming languages, such as FORTRAN and ALGOL 68 (Backus 1978, 615). The ENIAC was rewired in 1947–48 to become a centrally programmed sequential machine, mainly due to von Neumann’s influence (De Mol, Carlé, and Bullynck 2015, 1017). The EDSAC, completed at Cambridge (England) in 1949, was built on the model of the EDVAC and led, in 1951, to the first book spreading a real experience of programming by Wilkes, David Wheeler (1927–2004), and Stanley Gill (1926–1975), which was so influential both in England and in the USA during the 1950s.¹⁵ This structure had already been the basis of Babbage’s general plan of the Analytical Engine in 1840, as it was regularly recalled *a posteriori* by the first computing scientists, in their effort to find paternity for their nascent discipline¹⁶ (Hartree 1949b, 56; Turing 1950, 439). It can be recognized, with distinct units functioning independently, and Babbage’s vocabulary coming from factory terminology:

- the store where numbers or variables were located, kept its name before being thought dynamically and renamed the memory
- the mill, where the operations took place, corresponded to the Central Arithmetical Part (CA in the Draft)
- barrels and punched cards corresponded to the Central Control Part (CC)
- the rack with its punched cards corresponded to the input-output devices.

This computer structure also inherited the separation between operations and numbers, deeply experienced all along the 19th and 20th centuries with both analog and digital computing machines (see section 4). Von Neumann was completely aware of manual techniques and current computational equipment. He designed the EDVAC as a calculator, “a machine to automate mathematical calculation,” answering directly to the present social needs in

14 Contemporaneous authors referred to it as to a collective work, by von Neumann with Eckert and Mauchly, the main engineers who designed and built the ENIAC, and afterwards created the EMCC (see section 4.2; Wilkes 1985, 109).

15 See also the contributions of Borrelli and Priestley in this volume.

16 Hartree quoted Aiken as asserting that with his ASCC machine, “Babbage’s dream comes true.”

computing, and focusing on the means to take full advantage of its high speed (Priestley 2011, 133). Vacuum tubes allowed to build devices for arithmetical operations, and some usual other ones—the square root for example—as specific circuits assembling electronic elementary elements. Moreover, the machine code gave the same binary form of expression to numbers and instructions, which could be kept in the same store.

The invention of the stored program now characterizing modern computers appeared revolutionary. It was designed as a pragmatic response to the slow-down produced by the external presentation of data and instructions, which ruined the considerable saving in computing time induced by the high speed of electronic devices (Priestley 2011, 123–26). It also prompted the building of technical means of storing instructions and data, as well as intermediary results, inside the machine—well beyond just the ENIAC accumulators. This would become a major problem for the following decades. But above all, since stored program machines eliminated any human intervention in the whole computation process, every step of the sequence of operations had to be thought through and written in advance. Programmers had to adopt the “machine’s eye view” (Hartree 1949b, 92), so as to make explicit what remained intuitive in human computations, not only to perform them, but to prevent errors that the slightest lack of instruction could induce. So, inside the machine, the division of mental labor was now extreme, encompassing not only the steps of usual arithmetical operations, but new operations related to their execution by the machine, such as indicating the locations of data and their displacements, notably between the arithmetical unit and the control unit, with their subsequent modifications of addresses. Only this complete set of such instructions would ensure that computations would be carried out blindly by fully automatic machines, which would ultimately deliver the expected numerical solution. Thinking about meaning was definitely up to the team of machine and program designers. No interpretation of computations occurred during the whole process, since any human intervention was excluded. However, as Hartree admitted easily, it remained very difficult to foresee “all the unexpected things that might occur in the course of the calculation” (Hartree 1949b, 93).

5.2 The Inadequacy of This Culminating View of the Division of Mental Labor in Programming

Planning a computation on a computer presupposed to make explicit each of its steps as a sequence of orders, each of them expressing an operation. In such a trend of thought, programming was initially conceived to specify the sequencing of operations that has to be carried out all along the computing process, defined as “a sequence of orders for performing some particular calculation” (Wilkes, Wheeler, and Gill 1951, 1). This view was particularly suitable

for machines with von Neumann architecture. These orders had also to be previously coded so as to be transmitted to the machine. At first glance, it could appear to be a fairly easy task, even if laborious: the programmer had to write, line after line, the sequence of orders to be transmitted to the machine for evaluating a numerical solution for a problem. From this still top-down perspective, only this transmission from humans to machines was considered. Because of the main source of problems at the time, the process was often observed as a translation from mathematical language in machine language through coding in a binary system. And this view was reinforced by the fact that “many computers were nearly completed before anyone even established how they would load instructions into the storage element to get the problem started” (Adams 1955, 75). One therefore needed programming experience to understand that a program was not a simple linearly written sequence of orders.

To automatize the whole sequence of operations required the inclusion of more than just the sequence of corresponding orders, because all the elements of computations could not be fixed beforehand. As previously indicated, besides arithmetical operations, it was required to locate the storage of data and orders, and specific operations had to indicate their transfer between storage and arithmetical unit (CA). Aiken already noticed in 1946 that “no longer does each line of coding correspond to a single operation of the machine” (Priestley 2011, 159). But instructions of transfer did not suffice for coding special stages of computations requiring a choice, such as conditional and unconditional jumps, as well as iterative processes, with their loops representing repetitive sequences of orders. Already for Turing, “the sequence of orders can [be] divide[d] at various points, continuing in different ways according to the outcome of the calculations to date. We also wish to be able to arrange for the splitting up of operations into subsidiary operations” (Turing 1945, 11). Addresses and instructions would need to be modified during the process, especially for iterative procedures where the number of iterations was not given in advance.

In 1947, Goldstine and von Neumann presented the methodology of programming as they conceived it for the IAS machine (1945-51-58; see section 4.2). A previous step was what they named “planning.” As on the differential analyzer, it corresponded to the work carried out by the mathematician to reduce a given problem to a system of equations, specifying their conditions of validity. The main part, “coding,” had to explore how specific orders for machine operations—transfer, substitution, repetition—modified the purely sequential way of thinking the computational process. The view of programming changed radically, from a simple reading of successive given orders, to an evolutive course where data and instructions could be modified during the computational progress. Since they were written and stored in the same

way, they could be submitted to the same operations. So, “the relation of the coded sequence to the mathematically conceived procedure of numerical solution is not a static one, that of a translation, but highly dynamical” (Goldstine and von Neumann 1947–48, 2). Goldstine and von Neumann elaborated in great detail how flow diagrams were “schematics” best able to represent this dynamical feature. This specific feature of programming showed a first failure of Babbage’s view asserting that the principle of the division of mental labor could completely structure computing processes.

5.3 Subroutines as an Essential Rebuilding of Local Meaningful Structures

As the increasing storage capacities of computers allowed them to handle more complex problems, the size of programs grew exceedingly. This phenomenon took such proportions later that it was sometimes referred to as program obesity, contributing to a software crisis (Backus 1978, 614). Anyway, it became very difficult to make second readings of programs, either to check or to detect errors, even for those who designed them. The too fine cutting in elementary units became a real handicap, as many authors testified (Adams 1955; Dijkstra 1968), starting with Wilkes:

By June 1949 people had begun to realize that it was not so easy to get a program right as had at one time appeared... I was trying to get working my first non-trivial program, which was one for the numerical integration of Airy’s differential equation... the realization came over me with full force that a good part of the remainder of my life was going to be spent in finding errors in my own programs. Turing had evidently realized this too. (Wilkes 1985, 145)

Already in the days of manual computations, it was common to recombine parts of successive elementary operations, when they had to be repeated, and to call them routines, a word perhaps borrowed from current vocabulary. In 1930, Sadler (see section 3.4) named “routine” the entire task of the junior staff “girls” completing the packs of 61 “dumb computer papers” designed by Comrie (Sadler 2008, 21). Even if he did not use the term, Hartree initiated the use of subroutines on the differential analyzer when he identified parts of a diagram as reusable in others, and saved them to be reintroduced directly into more complicate set-ups of the machine. These partial diagrams were all the more important when he considered the differential analyzer could calculate tables of functions, so as to avoid the reading of graphics on material tables by operators (Hartree 1938, 351–57). This practice was also developed on the ASCC and the ENIAC, where the master-programmer organized sub-sequences of computations (Priestley 2017, 461). With stored-program machines, sub-routines took a more flexible coded form, prompting reflection on how to

introduce them directly into a program—once or multiple times—by their location addresses. Absent from the “Draft,” subroutines were central in Turing’s presentation of his Electronic Computer project—the Automatic Computing Engine (ACE)—at the National Physical Laboratory (Turing 1945, 34–35), and Goldstine and von Neumann also detailed the various cases. The resort to subroutines in programs became completely operational with the completion of the EDSAC in 1949 at Cambridge, where Hartree largely contributed to establishing the library of subroutines on punched tapes. Wheeler underlined the huge work it needed, and elaborated methods “for adjusting a subroutine so that it becomes correct in the set of locations in which it is put and used” (Wheeler 1952, 235b). So, the whole organization of certain problems could now be conceived as a master routine. The full title of the book where Wilkes, Wheeler and Gill published their enlarged experience on the EDSAC, *The Preparation of Programming for an Electronic Digital Computer, with Special Reference to the EDSAC and the Use of a Library of Subroutines* (1951), clearly showed the importance of subroutines in the programming process. It had a great impact also in the USA.

Of course, subroutines were themselves mechanized thanks to successive elementary operations, but they needed to be designed as meaningful local structures to help programmers manage their main intellectual work tool, namely programs, without which computers would be completely useless. With the master routine and subroutines previously coded, the task of coding the remaining part of the program became much lighter, even if the execution took more time than direct coding. Anyway, this practice of subroutines also reduced considerably the risk of errors in programs. As Wheeler noticed:

The subroutines can be more easily coded and then tested in isolation from the rest of the program. When the entire program has to be tested, it is with the fore knowledge that the incidence of mistakes in the subroutines is zero (or at least one order of magnitude below that of the untested portions of the program!). (Wheeler 1952, 235a)

As an economical issue, the search for the best balance between the time cost and the ease of programming had to be constantly negotiated when designing computers (PROGRAMme § 7), especially when they entered the commercial market. This aspect must never be forgotten, and Backus recalled it as one facet of the search for programming languages: “The cost of programmers associated with a computer centre was usually at least as great as the cost of the computer itself... In addition, from one-quarter or one-half of the computer’s time was spent in debugging” (Backus 1981, 28).

Programming languages extended the multiple possible occurrences of subroutines in a program. With FORTRAN II in 1958, they could call them freely. Further, when Allen Newell (1927–1992) and Herbert Simon (1916–2001)

conceived the Logic Language of the Logic Theorist in 1956, what was occasional became usual practice (Priestley 2017, 464). More essentially, subroutines could also be called from other subroutines, and introduced by recursive calls, so that “organizations of processes that [were] often hierarchical, iterative and recursive” could be expressed (Priestley 2017, 463–64). So, the hierarchy of subroutines introduced complexity in programming, where the execution of the master process was defined dynamically, depending heuristically on previous results. The major role of subroutines, reintroducing meaningful structures at a local level, therefore also showed that programming could not be reduced to a process dividing mental labor down into minimal, meaningless units.

5.4 From Machine Code to Programming Languages: A Return to Meaning

As electronic high-speed machines worked in binary systems, programs needed first to be written in the machine code, which can be considered as the finest possible breakdown of arithmetical operations and control orders into elementary units. Coding was a particularly laborious task, carried out in fact by programmers, as there was no defined separation neither between tasks nor people (De Mol and Bullynck, 2022, 156). As early as 1946, Arthur Burks (1915–2008), Goldstine and von Neumann introduced a more readable list of “symbols with a mathematical meaning” (Burks, Goldstine, and von Neumann 1946, 42)—in fact, abbreviations for the operations of the computer. This list was then taken up and extended by their followers. Goldstine and von Neumann also underlined that “coding [was] not a static process of translation but rather the technique of producing a dynamic background to control the automatic evolution of a meaning” (Goldstine and von Neumann 1947–48, 2). So, more than the meaning of arithmetical operations, programmers had to understand the meaning of the machine operations, what Mark Priestley named “the semantics of programming” (Priestley 2011, 177).

As programming was a very time-consuming task, researchers undertook the development of automatic programming, or rather automatic coding, structuring these symbolical abbreviations so as to obtain their automatic translation into machine code. A first step in that direction came from Wheeler on the EDSAC; he conceived initial orders and interpretative subroutines as two alternative techniques to correctly insert subroutines in any program (PROGRAMme § 6). In the first case, their correct addresses were automatically calculated, or translated, before the program execution, whereas in the second case, this translation was performed during the execution by additional pseudo-orders (Wheeler 1950). These two trends of thought inspired generalized approaches to entire programs. As interpretative subroutines, interpreted codes, named pseudo-codes, were transformed into

machine code during the program execution, such as the Short Code implemented on the UNIVAC in 1950, until the formulation of the Logic Language of Newell and Simon in 1956, first implemented on the JOHNIAC. But Wilkes' synthetic orders were transformed in advance. The *Symposium on Automatic Programming for Digital Computers* (1954) collected reference material on the subject, and intended to accelerate the automatization of programming. As pseudo-codes needed notation, the interpretation of their orders in machine code became known as translation (Priestley 2011, 188). Mathematics—especially numerical analysis—was deeply involved in computer practices, and mathematical notations were very familiar in the environment of computers, so they were more and more often chosen to express these pseudo-codes, which were on their way to becoming programming languages. When Alick Glennie (1923–2003), who first contributed to the library of subroutines on the EDSAC (Wilkes 1985, 209), presented his Autocode—often considered as the first compiler—for the Mark I at Manchester University, he insisted on the advantage of mathematical notations to make coding comprehensible even for the novice, and to make “automatic translation by the machine fairly simple to programme” (Glennie 1953, 6). Among the programming systems presented in 1954, Laning and Zieler used pseudo-codes in mathematical notations. And in the successfully commercialized programming language FORTRAN (1958), mathematical formulas were automatically translated into a sequence of basic instructions for the machine.

So, the illusion existing at least since Babbage that computations could be carried out without any occurrence of meaning radically failed, as computers could not work efficiently without programs, and programs needed to reintroduce both local structures and new notations, not only to be read by machines, but also to be understandable and manageable for programmers.

6. The New Situation of Actors on Computers

Reconstructing the design of computers since the 19th century, two categories of actors were described. On the one hand, mathematicians, astronomers, and physicists, who worked more and more collectively with engineers, and were really considered as effective actors. On the other hand, manual computers, considered as executing passively very elementary operations, and seemingly disappearing in the mechanization process.

In fact, this category of workers did not disappear, but their repetitive labor changed in nature: they no longer carried out repetitive parts of operations but were now devoted to serving the machine. A document edited in 1959 by the US Department of Labor listed a hierarchy of jobs, with a brief description, focusing on the education, training and experience needed, as well as on their characteristics. The list included such employments as: Card-Tape-Converter

Operator, Coding Clerk, Computing Analyst, Console Operator, Data typist Code, Electronics Mechanic for Computer, High Speed-Printer Operator, Programmer, Programmer Chief, Project Planner for Data Processing System, Supervisor for Data Processing System, System Analyst, and Tape Librarian (Mitchell 1959). So, when computers were commercialized, their running mobilized a host of specialized employments that were rarely mentioned in the literature.

As soon as computers were no longer prototypes and began to be commercialized, these actors diversified significantly, following the industrial mode of production. This diversification indicates that it would be a serious weakness in analysis to consider people involved in computers as humans in general. Notably, the practice of computers led to considering the importance of a specific group of them, the users, who were no longer the machine designers. They deserved particular attention, since both Babbage and the first computer designers saw any other human intervention only as error-prone. It became obvious that programming did not only consist of “optimizing the efficiency of computational process” (Dijkstra 1972, 860), but that it had to mediate communications between machines and these users. Consequently, programming languages were not universal—they had to be adapted to particular needs (PROGRAMme, § 7). As Adams underlined in 1955, the efficiency of a programming language depended on who would use it. The algebraic coding in Laning and Zieler was one of his examples that appealed to an occasional coder, while the Summer Session Computer at the MIT was only for training purposes.¹⁷ Before FORTRAN, commercial purposes were the focus of systems such as the A-2 and the B-Zero systems produced by Grace Hopper’s group at Remington Rand, or “Dr. Mitchell’s BIOR, intended for the professional business user” (Adams 1955, 77). This new position of computer users required considering the user not as a passive receiver of a technology already available, but as a full-fledged actor, thus renouncing the top-down perspective so “natural” for Babbage and others. E.W. Dijkstra (1930–2002) still deplored it as not really the case several decades later:

Considering, however, that the programming language is the bridge between the user and the machine—that it can, in fact, be regarded as his tool—it seems just as important to take into consideration “what Man can think.” (Dijkstra 1985, 216)

But it is not certain this view could be easily heard and developed, given that it was facing several centuries of attachment to the top-down approach, supported by the separation between liberal arts and mechanical arts, and reinforced by the enthusiastic view of computers as electronic brains (Berkeley 1949), and the context of their achievement.

17 See Mark Priestley’s paper in this volume.

7. Conclusion

What happened to the division of mental labor from Babbage's perspective to this stage of development of computers? In the analog instruments and machines discussed here, mechanizing the integration process was based on an elementary mechanism, the integrator, included in more complex set-ups. For digital computing, however, since De Prony's planning, the process was thought and practiced manually before any attempt was made to mechanize the whole task. In both cases, the computing process was broken down into elementary units. These units were initially arithmetical operations, until electronic elements allowed them to be built as special networks represented by block symbols, so that the division of mental labor seemed to have reached its ultimate stage. Moreover, on electronic machines, the whole computing process became invisible to any outside observer. Combined with the separation between more and more complex systems of data and the computation mechanisms, this invisibility covered up the meaning of these computations from the eye, so that this whole process appeared as completely reified.

Babbage's failure to realize not only his Analytical Engine but even his Difference Engine highlights the fact that the construction of computers required production conditions which went far beyond the sole intellectual scope of their design. As he already foresaw—but without suspecting the extent—the machine had to deal with an extensive amount of numerical data, and to provide results for experimental physics and social needs. So, the perspectives were very different: whereas Babbage set out to mechanize algebraic calculations conceived as expressing operations of mind, computing machines became necessary to obtain approximate numerical solutions when exact solutions were mathematically unreachable. They also required increasingly considerable resources, further amplified by the Second World War, namely enormous funding, and institutionalized collaborations between universities, industry, and defense. These conditions continued to exist during the Cold War, in a more open way toward the civilian world. From this period onwards, scientists did not occupy the privileged position Babbage wished, advising government to coordinate industrial production in analogy to the laws of nature. Instead, they were integrated into research teams, where they shared decision-making power with physicists and engineers, as well as those—industrialists, military, and others—at the origin of the problems they had to deal with.

However, high-speed electronic machines raised programming as a new significant problem, showing that the top-down perspective Babbage considered with the division of mental labor was too simplistic to account for the way computers work. It was not enough to adopt the “machine's eye,”

to add machine operations to arithmetical ones, and to write programs line by line in a form adapted to this principle. Programs were first prepared for the machine, but they also needed to be manageable, and therefore understandable, by programmers. From this perspective, from subroutines to programming languages, new thought structures were introduced, and the issue of meaning made its comeback. Programming languages need to be thought of from a pragmatic view of language, not only considering its syntax and semantics. Their meaning has to coordinate the expectations of the various actors concerned by computers, including users. Such a coordination requires rehabilitation of the intervention of other actors beside the initial designers of computers. Whatever be its autonomy, a computer remains a machine, designed and built to serve specific interests, and the computer operates according to which interests are selected.

I would like to thank Maarten Bullynck and Arianna Borrelli for their reading and precious advice during the preparation of this paper.

References

- Adams, Charles. 1955. "Developments in Programming Research." *Proceedings of the Eastern Computer Conference*, November 7–9, 1955: 75–79.
- Airy, George. 1842. "Letter to the Chancellor of the Exchequer, 1842, sept. 19." Airy manuscripts, Folios 65–66. Documents Archives, Cambridge University Library.
- Andoyer, Henri. 1915. *Nouvelles tables trigonométriques fondamentales, contenant les valeurs naturelles des lignes trigonométriques de centième en centième du quadrant avec vingt décimales, de neuf en neuf minutes avec dix-sept décimales, et de dix en dix secondes avec quinze décimales*. Paris: Hermann.
- Anonymous, 1821. "Inaugural Address." *Memoirs of the Astronomical Society* 1: 1–7.
- Ashworth, William. 1994. "The Calculating Eye: Baily, Herschel, Babbage and the Business of Astronomy." *The British Journal for the History of Science* 27: 409–41.
- Aspray, William. 1989. "Early Computing and Numerical Analysis at the National Bureau of Standards." *IEEE Annals of the History of Computing* 11, no. 1: 3–12.
- Aubin, David. 2009. "Observatory Mathematics in the Nineteenth Century." In *Oxford Handbook for the History of Mathematics*, edited by Eleanor Robson and Jacquelin Stedall. 273–98. Oxford: Oxford University Press.
- Aubin, David, and Catherine Goldstein, eds. 2004. *The War of Guns and Mathematics: Mathematical Practices and Communities Through World War I in France and its Western Allies*. Berlin: Springer.
- Babbage, Charles. 1830. *Reflections on the Decline of Science in England, and Some of its Causes*. London: B. Fellowes.
- . 1832. *On the Economy of Machinery and Manufactures*. London: Charles Knight.
- . 1864. *Passages from the Life of a Philosopher*. London: Longman.
- Babbage, Henry. 1910. "Babbage's Analytical Engine." *Monthly Notices of the Royal Astronomical Society* 70: 517–26, Erratum 645.
- Backus, John. 1978. "Can Programming Be Liberated from the von Neumann Style?" *Communications of the ACM* 21, no. 8: 613–41.
- . 1981. "The History of Fortran I, II and III." In *History of Programming Languages Conference*, edited by Richard Wexelblatt, 25–45. Cambridge, MA: Academic Press.
- Barbin, Adeline. 2020. "Les relations entre la société et les techniques. Comment la question se pose-t-elle et que peut nous en apprendre la philosophie?." In *Rencontre autour de*

- Bruno Bachimont : *La technologie nous fait-elle savoir et penser autrement?*, edited by Michaël Crevoisier, Fabien Ferri, and Carol Widmaier, 97–107. Besançon: Presses Universitaires de Franche-Comté.
- Berkeley, Edmund. 1949. *Giant Brains: Or Machines That Think*. New York: Science Editions.
- Birkhoff, Garrett. 1980. "Computer Developments 1935–1955, as Seen from Cambridge." In *History of Computing in the 20th Century*, edited by Nicholas Metropolis, Jack Howlett, and Gian Carlo Rota, 21–30. New York: Academic Press.
- Brown, Ernest. 1919. "Preface." In *Tables of the Motion of the Moon*, edited by Ernest Brown and Henry Hedrick, Volume 1, v–xii. New Haven: Yale University Press.
- Bullynck, Maarten. 2018. "Programming Men and Machines. Changing Organization in the Artillery Computations at Aberdeen Proving Ground (1916–1946)." *Revue de Synthèse* 139, no. 3–4: 237–58.
- Burks, Arthur, Herman Goldstine, and John von Neumann. 1946. *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*. Princeton: Institute for Advanced Study.
- Bush, Vannevar. 1931. "The Differential Analyzer. A New Machine for Solving Differential Equations." *Journal of the Franklin Institute* 21: 44–48.
- Cannon, Walter. 1964. "Scientists and Broadchurchmen: An Early Intellectual Network." *Journal of British Studies* 4, no. 1: 65–88.
- Comrie, Leslie. 1925. "The Application of Calculating Machines to Astronomical Computing." *Popular Astronomy* 3: 243–46.
- . 1928. "On the Application of the Brunsviga-Dupla Calculating Machine to Double Summation with Finite Differences." *Monthly Notices of the Royal Astronomical Society* 88, no. 5: 447–59.
- . 1932. "The Application of the Hollerith Tabulating Machine to Brown's Tables of the Moon." *Monthly Notices of the Royal Astronomical Society*, 92: 694–707.
- Darwin, Charles. 1958. "Douglas Rayner Hartree (1897–1958)." *Biographical Memoirs of the Fellows of the Royal Society* 4: 103–16.
- Dauben, Joseph, and Christopher Scriba, eds. 2002. *Writing the History of Mathematics: Its Historical Development*. Basel: Birkhauser Verlag.
- Daylight, Edgar. 2015. "Towards a Historical Notion of 'Turing—the Father of Computer Science'." *History and Philosophy of Logic* 36, no. 3: 205–28. <http://dx.doi.org/10.1080/01445340.2015.1082050>.
- De Mol, Liesbeth, Martin Carlé, and Maarten Bullynck. 2015. "Haskell Before Haskell. An Alternative Lesson in Practical Logics on the ENIAC." *Journal of Logic and Computation* 25, no. 4: 1011–46.
- De Mol, Liesbeth, and Maarten Bullynck. 2018. "Making the History of Computing. The History of Computing in the History of Technology and the History of Mathematics." *Revue de Synthèse* 139, no. 3–4: 352–72.
- De Mol, Liesbeth, and Maarten Bullynck. 2022. "What's in a Name? Origins, Transpositions and Transformations of the Triptych Algorithm -Code -Program." In *Abstractions and Embodiments: New Histories of Computing and Society*, edited by Janet Abbate and Stephanie Dick, 146–68. Baltimore: Johns Hopkins University Press.
- Dijkstra, Edsger. 1968. "Go To Statement Considered Hateful (letter to the editor)." *Communications of the ACM* 11, no. 3: 147–48.
- . 1972. "The Humble Programmer." *Communications of the ACM* 15, no. 10: 859–66.
- . 1975. "Correctness, Concerns and, among Other Things, Why They Are Resented." *SIG-PLAN Notices* 10, no. 6: 546–50.
- . 1965. "Programming Considered as a Human Activity." In *Proceedings of the IFIP Congress*, edited by W. A. Kalenich, 213–18. Washington, DC: Spartan Books.
- Dubbe, John M. 1978. *The Mathematical Work of Charles Babbage*. Cambridge: Cambridge University Press.
- Durand-Richard, Marie-José. 2000. "Logic versus Algebra: English Debates and Boole's Mediation." In *A Boole Anthology: Recent and Classical Studies in the Logic of George Boole*, edited by James Gasser, 139–66. Dordrecht: Kluwer Academic Publishers.

- . 2010. "Planimeters and Integrals in the 19th Century, Before the Differential Analyzer." *Nuncius* 34, no. 1: 101–24.
- . 2016. "De la prédiction des marées: entre calcul, observation et mécanisation (1831–1876)." *Cahiers François Viète* 2, no. 8–9: 105–35. <https://journals.openedition.org/cahierscfv/2535>.
- . 2018. "Douglas R. Hartree (1897–1958): de l'analyseur différentiel à l'ordinateur." *Revue de Synthèse* 139, no. 3–4: 289–326. <https://halshs.archives-ouvertes.fr/halshs-01516339>.
- . 2020. "Fondements symboliques des pratiques algorithmiques en Angleterre (1801–1860)." In *L'Épistémologie du dedans: Mélanges en l'honneur de Hourya Benis-Sinaceur*, edited by Emmylou Haffner and David Rabouin. Paris: Editions Garnier.
- . 2022. "Historiographie du calcul graphique." In *Histoire du calcul graphique*, edited by Dominique Tournès, 463–523. Paris: Editions Cassini. <https://halshs.archives-ouvertes.fr/halshs-01516382>.
- , and Philippe Guillot. 2019. "From Poznań to Bletchley Park: The History of Cracking the ENIGMA Machine." *CIIT Lab Workshop on History of Cryptography*, Faculty of Electronic Engineering, Niš, Serbia, October 2017. <https://shs.hal.science/halshs-03949790v1>.
- Eckert, William. 1940. *Punched Card Methods and Scientific Computation*. Columbia University: Thomas J. Watson Astronomical Computing Bureau.
- Glennie, Alick. 1953. "Lecture Notes: The Automatic Coding of an Electronic Computer." Lecture delivered at the University of Cambridge. Computer History Museum, catalogue number 102653981. <https://www.computerhistory.org/collections/catalog/102653981>.
- Goldstine, Hermann and John von Neumann. 1947–48. *Planning and Coding of Problems for an Electronic Computing Instrument: Report on the Mathematical and Logical Aspects of an Electronic Computing Instrument*. Part II. Volumes 1–3. Princeton: Institute for Advanced Study.
- Grattan-Guinness, Ivor. 2003. "The Computation Factory: de Prony's Project for Making Tables in the 1790s." In *The History of Mathematical Tables: From Sumer to Spreadsheets*, edited by Martin Campbell-Kelly, Mary Croarken, Raymond Flood, and Eleanor Robson, 101–21. Oxford: Oxford University Press.
- Grier, David. 2003. "Table Making for the Relief of Labour." In *The History of Mathematical Tables: From Sumer to Spreadsheets*, edited by Martin Campbell-Kelly, Mary Croarken, Raymond Flood, and Eleanor Robson, 265–92. Oxford: Oxford University Press.
- Hartree, Douglas. 1938. "The Mechanical Integration of Differential Equations." *The Mathematical Gazette* 22: 342–63.
- . 1940. "A Great Calculating Machine: The Bush Differential Analyser and its Application in Science and Industry." *Proceedings of the Royal Institution* 3: 151–94.
- . 1949a. *Differential Analyser*. Ministry of Supply Monograph. Hartree Fund C4. Manchester: National Archives on the History of Computing.
- . 1949b. *Calculating Instruments and Machines*. Urbana: The University of Illinois Press.
- Hyman, Anthony. 1983. *Charles Babbage, Pioneer of the Computer*. Princeton: University of Princeton Press.
- Lacroix, Sylvestre. 1819. *Traité du calcul différentiel et du calcul intégral. 3ème partie. Traité des différences et des séries*. Paris: Courcier.
- Laplace, Pierre Simon. 1798–1825. *Traité de Mécanique Céleste*. Five volumes. Paris: J. B. M. Duprat.
- Marx, Karl. 1867–1894. *Das Kapital: Kritik der politischen Oekonomie*. Three volumes. Hamburg: Otto Meissner.
- McCulloch, Warren. 1949. "The Brain as a Computing Machine." *Electrical Engineering* 6: 492–97.
- Mitchell, James. 1959. *Occupations in Electronic Data-Processing Systems*. Washington, DC: US Employment Service, US Department of Labor.
- Morrell, Jack, and Arnold Thackray. 1981. *Gentlemen of Science: Early Years of the British Association for the Advancement of Science*. Oxford and New York: Clarendon Press.
- Norberg, Arthur. 2003. "Table Making in Astronomy." In *The History of Mathematical Tables: From Sumer to Spreadsheets*, edited by Martin Campbell-Kelly, Mary Croarken, Raymond Flood, and Eleanor Robson, 177–207. Oxford: Oxford University Press.

- Olley, Allan. 2018. "A Task That Exceeded Technology: Early Applications of the Computer to the Lunar Three-body Problem." *Revue de Synthèse* 139, no. 3–4: 352–72.
- Peacock, George. 1830. *A Treatise on Algebra*. Cambridge: Rivington and Whittaker.
- Pestre, Dominique. 2006. *Introduction aux Sciences Sociales*. Paris: La Découverte.
- PROGRAMME Collective. (Forthcoming). "Notations. There is No Escape." In *What is a Computer Program?* ANR project PROGRAMme, directed by Liesbeth de Mol.
- Priestley, Mark. 2011. *A Science of Operations: Machines, Logic and the Invention of Programming*. London: Springer.
- . 2017. "AI and the Origins of the Functional Programming Language Style." *Minds & Machines* 27: 449–42.
- Rees, Mina. 1987. "The Computing Program of the Office of Naval Research, 1946–1953." *Communications of the ACM* 30, no. 10: 832–48.
- Sadler, Donald. 2008. *A Personal History of the H. M. Nautical Almanac Office. 30 October 1930–18 February 1972*, edited by George A. Wilkins. United Kingdom Hydrographic Office.
- Segal, Jérôme. 2003. *Le Zéro et le Un, histoire de la notion scientifique d'information au 20ème siècle*. Paris: Editions Syllepse.
- Smith, Adam. 1776. *An Enquiry into the Nature and Causes of the Wealth of Nations*. London: W. Strahan and Cadell.
- Stewart, Dugald. 1792–1827. *Elements of the Philosophy of the Human Mind*. Three volumes. London and Edinburgh: Strahan & Cadell and Creech.
- Stewart, Dugald. 1835. *Lectures on Political Economy*, edited by Sir William Hamilton. Edinburgh: T & T. Clark.
- Swade, Doron. 1996. "Charles Babbage's Difference Engine n° 2. Technical Description." *Science Museum Papers in the History of Technology* 4.
- Tournès, Dominique, ed. 2022. *Histoire du calcul graphique*. Paris: Editions Cassini.
- Trogemann, Georg, Alexander Y. Nitusssov, and Wolfgang Ernst, eds. 2001. *Computing in Russia: The History of Computer Devices and Information Technology Revealed*. Braunschweig: Vieweg (translated by A.Y. Nitusssov).
- Turing, Alan. 1936–37. "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society* (1) 42: 230–65 and (2) 43: 544–46.
- . 1945. "Proposed Electronic Calculator." *Minutes of the Executive Committee of the NPL*. AlanTuring.net. Accessed October 2024. www.alanturing.net/turing_archive/archive/index/aceindex.html.
- . 1950. "Computing Machinery and Intelligence." *Mind* 59, no. 236: 433–60.
- Ure, Andrew. 1835. *The Philosophy of Manufactures, or an Exposition of the Scientific, Moral, and Commercial Economy of the Factory System of Great Britain*. London: Charles Knight.
- von Neumann, John. 1993 [1945]. "First Draft of a Report on the EDVAC." *IEEE Annals of the History of Computing* 15, no. 4: 27–75.
- Wheeler, David. 1950. "Programme Organization and Initial Orders for the EDSAC." *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 202, no. 1071: 573–89. <https://doi.org/10.1098/rspa.1950.0121>.
- Wheeler, David. 1952. "The Use of Sub-routines in Programs." *Proceedings of the Association for Computing Machinery National Meeting, Pittsburgh*: 235–36. <https://dl.acm.org/doi/10.1145/609784.609816>.
- Wilkes, Maurice, David Wheeler, and Stanley Gill. 1951. *The Preparation of Programs for an Electronic Digital Computer*. Reading, MA: Addison-Wesley.
- Wilkes, Maurice. 1985. *Memoirs of a Computer Pioneer*. Cambridge, MA: MIT Press.
- Wilkins, George. 2003. "The Making of Astronomical Tables in HM Nautical Almanac Office." In *The History of Mathematical Tables: From Sumer to Spreadsheets*, edited by Martin Campbell-Kelly, Mary Croarken, Raymond Flood, and Eleanor Robson, 295–320. Oxford: Oxford University Press.
- Williams, Michael. 1985. *A History of Computing Technology*. Englewood Cliffs: Prentice-Hall.

- . 2003. "Difference Engines: from Müller to Comrie." In *The History of Mathematical Tables: From Sumer to Spreadsheets*, edited by Martin Campbell-Kelly, Mary Croarken, Raymond Flood, and Eleanor Robson, 122–44. Oxford: Oxford University Press.

HUMANITIES COMPUTING

DIGITAL HUMANITIES

POCKET CALCULATOR

MEDIEVAL HISTORY

MAINFRAME COMPUTER

Mainframe Computer or Programmable Pocket Calculator? Calculation Tools and Practices of Computing in Medieval History (1960s–1980s)

Edgar Lejeune

From the 1960s to the late 1980s, a paradigmatic model for humanities computing was the collective study of large datasets using mainframe computers. From this model derived a characteristic social organization of the work and a related set of practices of computing. In the field of medieval history, several projects of that kind were conducted in Europe during that period, e.g., the pioneer international study of the 1427 Florentine *catasto* conducted by David Herlihy and Christiane Klapisch-Zuber. In the 1970s, the French medievalist Alain Guerreau strongly criticized this kind of computing practice and decided to turn his attention to another type of computing equipment: the programmable pocket calculator. This paper proposes a comparison between these two ways of computing in order to address two

related issues. How does a researcher choose a computation tool within an array of available devices? And how does their equipment shape their practice of computing? In asking so, this article aims at providing fuel for two major contemporary debates in the history of computing. First, what should be considered a computer within our historical approach to computing? And to what extent should automatization and programming functions be considered crucial characteristics for defining modern computers? Second, what is the history of computing the history of? Is it concerned with the history of people “using computers”? If so, what does “using” a computer historically mean?

“The computer is not one thing, but many different things, and the same holds true of computing.” (Mahoney, 1988)

In one of the rare texts dedicated to the history of pocket calculators, Haigh and Ceruzzi reproduced a 1983 photograph of the astronaut Sally Ride (1951–2012) aboard the Challenger shuttle. In the legend of that figure, the two authors insist on the presence of “three Hewlett-Packard programmable calculators floating next to her,” which, despite the presence of a computer for guidance, navigation and control, “saw heavy use by the crew” (Haigh and Ceruzzi 2021, 169).

This competition between computers and other digital computing equipment in the same scientific environment is not a specificity of that spatial mission in the 1980s. If we could go back in time and have a look, for example, at a quantitative historian’s office at the same period, we would also find various devices that would be meant to perform computing operations. Apart from “traditional” paper tools such as lists and tables, we might also find a computer terminal, connected to the closest mainframe computer center, boxes of punched cards, mechanical desk calculators, or... digital pocket calculators (Lejeune 2021).

To my knowledge, this historical reality has been barely dealt with by the scientific literature on the history of computing. In order to fill that gap, the study of digital pocket calculators' practices appear as wonderful matter for research, and some specialists have already underlined their critical role in the evolution of early computing practices.

In the 1970s, the increasing number of pocket calculator sales and the rapid drop in their prices has for example been considered "the first indication" of the personal computing revolution, both in the United States and in the Soviet Union (Tatarchenko 2019; Haigh and Ceruzzi 2021, 169). Thereby, the commercialization of the first programmable calculators could be considered a turning point in the history of early personal computing. From the first programmable desk calculator, such as the 35 kilogram Olivetti Programma 101 (1965), the Monroe Epic (1965), or the Soviet ISKRA 123 (1973), to the marketing of the first programmable pocket calculator by Hewlett-Packard in 1974, the practices of computing were strongly affected by the irruption of these new machines onto the market.

Much more affordable than both mainframe computers and microcomputers, these devices allowed users to "store and execute short programs" (Haigh and Ceruzzi 2021, 169), and thus created an incredible opportunity for onlookers to access a powerful computing machine. At a time when mainframe computers were not available for a lot of people because of the heaviness of the human-machine interface they implied, the simplicity of pocket calculators and the liveliness of their users' community offered strong advantages to self-learn programming outside computer centers.

In the mid-1970s, the French medievalist Alain Guerreau (1948–) was one of these onlookers. In 1976, he decided to purchase a Texas Instrument TI-58 with the aim of performing basic statistical analysis as well as programming multivariate statistical procedures (Guerreau, 1981, 1982). In doing so, Guerreau did not just make a pragmatic choice toward the less expensive machine on the market. He also claimed to reject the paradigmatic model of humanities computing at the time, characterized by:

- the use of a mainframe computer, usually installed in an inter-university computing center
- the edition of data onto punched cards and/or magnetic tape
- the constitution of inter-disciplinary research teams, composed of computer specialists and humanities scholars.

This position defended by Guerreau is a fascinating starting point from which to raise major questions for both the history of science and the history of computing. How does a researcher choose a computation tool from an array of available devices? Besides the obvious financial criteria, what were the

arguments supporting their decision? And to what extent did these different devices complete each other, or compete with each other?

Guerreau's original choice also raises another set of key issues for our field of research. How did his equipment choice shape his practice of computing? What were the main differences between Guerreau's workflow and that of others, based on the uses of a mainframe computer? And thus, how is the choice of a given computing device a crucial element in the study of a given computing culture?

In order to answer these questions, the methodology I deploy will consist of comparing Guerreau's practice of computing with the results of another case study extensively analyzed in my PhD dissertation (Lejeune 2021): the study of the Florentine *catasto* of 1427, launched by the American medievalist David Herlihy (1930–1991) and the French medievalist Christiane Klapisch-Zuber (1936–2024) in the second half of the 1960s. I will show how it embodies the paradigmatic model of computer-assisted studies in the humanities described by Guerreau.

In order to develop this analysis, I will rely on three different types of archive material: institutional and personal archives of historians using computers in the 1970s and 1980s; published articles concerning computer-assisted projects in medieval history; and interviews of actors recorded between 2016 and 2018. These archive materials will help address my questions on the basis of the actors' motivations and decisions, and avoid caricatural and deterministic conclusions about the relationship between a given piece of computing equipment and a given practice of computing.

I will first focus on the rationales that led these two groups of actors to choose different computing devices. I will then analyze the impact of these choices on the computing cultures they developed, considering the idea that a given scientific culture is essentially shaped by the type of texts and intellectual technologies the actors put into play.

1. The Reasons for a Choice: Mechanical Calculators, Mainframe Computers, or Programmable Pocket Calculators

In this section, I will expose the logic that led historians to choose their computation tools in the late 1960s and early 1970s. In order to do so, I will first present the reasons that led some historians to work with mainframe computers in the late 1960s, in a context where mechanical desk calculators were widely used by actors interested in statistical applications to historical data. I will then describe the model that arose from the first computer-assisted

projects, in order to explain why scholars like Guerreau chose to turn their attention toward a more personal practice of computing.

1.1 Humanities Computing and Data Processing in the Late 1960s: Two Rationales for a Unique Computing Device

In 1966, the American medievalist David Herlihy proposed to Emmanuel Le Roy-Ladurie (1929–) and Fernand Braudel (1902–1985), two of the leading figures of the *Annales* school (Burke 1990), that they should join forces in order to study an archive monument of the history of late medieval Italy: the Florentine *catasto* of 1427. Composed of more than 60,000 tax declarations of every household of the Florentine territory, this source documents the life of more than 260,000 people of the late medieval period, and thus constitutes priceless material for medievalists. In one of the oldest letter kept in the archive, dated November 1966, Herlihy defined his ambition as follows:

The project I have been working on is to prepare an “edition” of the Catasto for the computer. This edition, once made, could be cheaply reproduced and utilized by scholars in any part of the world. They could, in other words, “program” the data in any way they wanted, according to their own research goals. Eventually, it seems to me, many such great statistical sources ought to be edited for the computer, so that entire libraries of such data could be accumulated.¹

Similar rationales supporting the choice to use computers were, at the time, quite common in the humanities. Since the late 1940s and the famous *Index Thomisticus* enterprise conducted by the Jesuit father Roberto Busa (1913–2011), using computers in order to produce a new type of scholarly instruments was a predominant motivation (Hockey 2004; Jones 2016; Wenin 1972).

In medieval history, the first attempts began in the early 1960s. At the CETEDOC, the computer center of Louvain Catholic university, the Belgian philologist Paul Tombeur (1936–) and his medievalist colleague Léopold Genicot (1914–1995) were for example engaged in the publication of computer-generated concordances and indexes of all literary texts produced in Belgium before 1200 (Genicot 1974; Tombeur 1970). At the University of Nancy, one of the first important centers for humanities computing in France, a team composed of linguists, medievalists and mathematicians developed an “edition for the computer” of charter documents produced before 1250 (Créhange and Fossier 1970; Fossier 1978). Many similar projects were also developed at the time in Italy, Spain, the United Kingdom, the United States,

1 Letter by Herlihy, November 20, 1966, EHESS CRH CKZ 1–13. Fond Christiane Klapisch-Zuber, Humathèque Condorcet, Aubervilliers (France). For details and references to the archival material see Lejeune (2021), page 45 in particular.

Germany, and the Netherlands (Burton 1981; de Tollenaere 1977; Wenin 1972; Wisbey 1963; Scafati 1977).

For this very reason, the early history of “humanities *computing*” has often been described rather as “an era of humanities *data processing*” (Jones 2016, 17) or “literary and linguistic computing” (Burnard 2012). By saying so, commentators acknowledge the fact that actors have considered computers primarily as text-based information retrieval tools, which could replace older and more “traditional” forms of scholarly instruments, e.g., printed indexes, concordances, or dictionaries (Lejeune 2024). Following that narrative, humanities scholars mobilized some specific functions of computers—counting, comparing, sorting, or identifying sequences of characters—and the possibility computers offered of easy processing of reproducible magnetic stored information was considered the main motivation for their uses.

However, that narrative does not deal with the fact that, at the time, the vast majority of calculators’ applications in history originated from the ambition to enhance quantitative methodology (Sula and Hill 2019; Crymble 2021). In the late 1940s and early 1950s, the first uses of electromechanical calculators were developed within the historical discipline with the aim of applying statistical methods to the study of massive electoral, demographical, or socio-economic sources.² Contrarily from the type of project developed by Busa, Tombeur, Genicot or Fossier, these historians were not trying to produce scholarly instruments. They were rather trying to find ways to automatize the application of statistical procedures to the huge tables produced by the administrators of the past, in a context where repetitive and simple mathematical operations represented the great bulk of their work.

Despite Herlihy’s will to use computers in order to create a new type of scientific edition, his choice to use a mainframe computer also fell into this statistic-related logic. Two striking elements of the definition of his project are of great interest to understand the latter. First, Herlihy insisted on the appropriateness of computers for dealing with the *catasto*, which he labeled as one among the “great statistical sources.” In his 1966 letter, Herlihy declared: “It really is the kind of massive, satisfactorily accurate survey which computers were designed to handle.”³

Similar kinds of assertion were in fact very common within the quantitative history literature in the 1950s and 1960s. They originated from a longer tradition of justifying the application of statistics to historical studies by insisting on some specific characteristics of the archive materials, i.e., their

2 Some important examples of such applications in France and in the United States are Aydelotte (1954); Bailyn and Bailyn (1959); Curti (1959); Daumard and Furet (1959); and Owsley and Owsley (1949).

3 Letter by Herlihy, November 20, 1966, EHESS CRH CKZ 1–13. Fond Christiane Klapisch-Zuber, Humathèque Condorcet, Aubervilliers (France).

size or the accuracy and comparability of the information they contain (Furet 1971; Lejeune 2021). In that context, computers thus appeared as an incredible means of processing the *catasto*'s information.

Then, mainframe computers also appeared to these historians as wonderful machines for applying statistical procedures. In a 1969 article, Christiane Klapisch-Zuber, the principal collaborator of Herlihy on the *catasto* survey, in charge of all the operations of the project in France, claimed:

...the sheer size of the document has discouraged comparative analysis, and the difficulty of consistently relating demographic information to economic data has almost always prevented exploitation of the most interesting aspect of the document: the simultaneous recording of goods and people. Only the computer can systematically correlate such a mass of information. (Klapisch-Zuber 1969, 1316)⁴

This assertion shows how much the choice toward a mainframe computer relied, in Klapisch-Zuber and Herlihy's minds, on the dimension of the source they were dealing with. Indeed, no other machines were at the time powerful enough to deal so quickly with the application of statistical procedures to tables composed of more than a thousand rows.

The reasons that led these historians to choose mainframe computers were incredibly similar to those developed at the time by statisticians. In a 1966 article, the British statistician Frank Yates (1902–1994) argued that computers were a “second revolution” in statistics, after the one that derived from the massive uses of mechanical desk calculators in the early 20th century (Yates 1966). At the time, Yates was doing research in biometrics with the help of a mainframe computer, an Elliott 401, at the Rothamsted Experimental Station. He was also engaged in the development of computer science in Great Britain as president of the British Computer Society, where he succeeded Maurice Wilkes (1913–2010).

In order to show why computers were very different from their technological ancestors, he first claimed in his 1966 article that they could be a solution for specialists to analyze huge datasets as a whole, in a context where mechanical desk calculators compelled to work on samples because of the time-consuming workflow that they implied. Then, he alleged that computers, thanks to the possibility of programming a sequence of operations, enabled applications of new types of statistical procedures, e.g., the computation of errors through co-variance analysis, or the research of unpredictable multifactorial relations between phenomena with the help of simulation and/or multivariate analysis.

The similarity between the statisticians' and the historians' rationales with regard to the use of mainframe computers shows that data processing was

4 All translations from the French original are by the author.

not the only reason for Herlihy to require the use of computers. The great methodological jump enabled by the possibility these machines offered of performing high-level statistical procedures, and of easily multiplying them, was critical to improving the methods of quantitative history.

In the specific case of the *catasto* survey, it made the comparative analysis of both demographical and socio-economical information of more than 260,000 Florentine taxpayers possible. It then enabled Klapisch-Zuber and Herlihy to publish a book in 1978, considered by both demographers and medievalists as a milestone for the studies of late medieval Italy (Lejeune 2021).

1.2 A Paradigmatic Model of Scientific Organization in the History of Computing

As I have shown above, mainframe computers were chosen by Herlihy and Klapisch-Zuber mainly because of their ambition to exhaustively analyze an incredible amount of historical information. In this section, I will demonstrate how, from this combination of characteristics (mainframe computer + huge archive material), derived 1) a characteristic social organization of the work and 2) a related set of edition practices of the data. As I will show in section 1.3, these elements will be later criticized by scholars like Guerreau and were then decisive in his decision to engage with other computing equipment.

The principal characteristic of a project like the one of Herlihy and Klapisch-Zuber relied on the incredible amount of work needed in order to edit the data onto magnetic tapes. In order to understand the modalities of that process, and thus the social organization implied, it is here necessary to quickly specify the material context in which this set of operations took place. Schematically, the production of the punched card file was done in four stages, each of which corresponded to a transfer of data storage.

The first stage consisted of the production of microfilms of certain parts of the *catasto*, as they enabled historians to work outside the Florence archives collections. The second stage consisted of recording some selected information from the Florentine declarations onto a coding sheet (see Figure 1).

These documents, very similar to two tables with 80 columns (see section 2.2.), were then sent to punching companies, which were responsible for transforming these sheets into a set of coupled 80-column IBM punched cards (Lejeune 2021). These cards, once returned to the École Pratique des Hautes Études (EPHE), were then stored and, above all, classified in a 30,000-card filing cabinet, so that they were not mixed up with each other.

The last step consisted of a final change of medium: the punched cards were input into the mainframe computer and recorded onto magnetic tapes. This last operation was absolutely crucial. On the one hand, it allowed a more

Yet, the most important actor of this part of the study was Michel Demonet, a young mathematician, who was just awarded a Diplôme d'études approfondies (DEA) in statistics under the direction of the French mathematician Jean-Paul Benzécri (1932–2019). In the 1970s, Demonet was one of the key figures in the computerization of historical research in France, after he was appointed by Benzécri in 1964 to work toward the application of computers with the contemporary historian Antoine Prost (1933–) (Prost 1974). Demonet then collaborated with linguists and historians from the lexicometry laboratory at the L'École normale supérieure (ENS) Saint-Cloud (Lejeune 2021; Loiseau and Léon 2016) before he was hired as a regular collaborator at the VIth section of the EPHE, where he was in charge of another computer-assisted international project, the *Statistique Générale de la France* (Béaur and Marin 2011).

His role in the *catasto* survey was pivotal because of two connected elements. First, he was responsible for solving a major material problem in the first years of the study, that is, to gain access to a computer center. In a context where these infrastructures were rare, and the privileged area of researchers in so-called exact sciences (Grossetti and Mounier-Khun 1995), Klapisch-Zuber and Demonet had to relocate their computing activities at least twice between 1968 and 1970. Around 1970, thanks to a funding strategy advised by Le Roy-Ladurie and Jacques Le Goff (1924–2014), their activities were finally transferred to one of the largest scientific computing centers in Europe: the Centre Inter-Régional de Calcul Électronique (CIRCE) of the Centre National de la Recherche Scientifique (CNRS), located near Paris, in Orsay.

Second, Demonet was also the only computer science referee on board with the French team of the project. He had then to assume several responsibilities, which started with both the writing of software programs imagined by Klapisch-Zuber and the debugging of programs sent by Herlihy's computer scientists, who were working on rather different, and usually more evolved, mainframe computers in the United States. He was also in charge of communicating with the operators of the computer center and of complying with the rules of these very particular institutions. For example, he had to predict the computing time of a given software program to be performed in order to respect the computing time they had been granted by the CNRS (Lejeune 2021).

Although no archive allows to better understand how these interdisciplinary translations were carried out, these pieces of information show that the division of labor at stake between historians, computer scientists and the computing center operators defined a model of computing that relied strongly on the choice toward mainframe computers. However, it was not directly the hardware solution at stake that shaped the division of labor I have described, but rather the social organization of academic computing at the time.

I have shown that using a mainframe computer in the humanities implied a strong dependence on academic infrastructures, laboratories and computing centers, entangled in complex and heavy administrative schemes (Mercouroff 2015; Connes and Perriquet 2022). Then, the mobilization of all these resources (human, equipment) entailed significant expenses, which increased the dependence on these academic institutions. These conclusions support those of other specialists who worked on the history of humanities computing (Jones 2016; Plutniak 2017). As we will see, they were also considered in the mid 1970s by actors as obstacles for the development of statistical-related methodologies in the humanities.

1.3 Guerreau's Choice of a Personal Calculator

In a 1981 article published in the journal *Les Annales ESC*, Guerreau proposed a statistical-supported analysis of the distribution of the mendicant orders in late medieval France (Guerreau 1981). At the time, Guerreau was a 33-year-old researcher at the CNRS, which he joined in 1978 after a short career as a secondary school teacher. However, he was not affiliated to any laboratory nor collective entities of research, but was mainly developing his skills in statistics and programming in an autodidactic fashion, using programming manuals and mathematical books. He was also very attentive to the computerization of other disciplines, especially sociology, a discipline in which Pierre Bourdieu (1930–2002) and his collaborators were massively using computing devices in order to perform multiple factor analysis (Duval 2013).

In that context, Guerreau argued in the first part of his 1981 article that the general evolution of the relations between statistics and social sciences caused so many problems since the late 1960s that it could be considered a moment of “crisis.” He notably identified two different and related issues he labeled as follows: “technical *versus* social” and “material *versus* intellectual.”

The first element he focused on is the radical transformation of computing resources imposed by the rapid progress of electronization. In the early 1970s, the development of new chip technologies, and especially the marketing of the metal-oxide semiconductor (MOS), caused rapid growth of micro-computing device production. As aforementioned in the introduction, the development of cheaper calculators thus appeared as a phenomenon of huge importance in the development of early personal computing, and could be considered “the first civilian counterpart to the Minuteman and Apollo programs of the 1960s” (Haigh and Ceruzzi 2021, 168). As a witness of that evolution, Guerreau acknowledged the wonderful growth in the computing power of individual machines, which from then on compete in speed and precision with mainframe computers.

Despite that major evolution, Guerreau deplored the fact that mainframe computers were still the dominant device for computing in the humanities, arguing that the social organization of computing was the main element of the crisis he described:

This technical aspect is of course closely linked to a social aspect—the cost of these instruments and their delicate operation have accentuated the division of labour by giving rise to a new specialist responsible for calculations and the handling of calculation tools. The perverse effect of this process lies less in the division itself than in the very dynamics of specialisation, which leads the new specialist to strengthen his autonomy by monopolising as best he can the know-how he possesses; thus the absurd social slope which leads to the separation of calculations from the rest of the research, which are carried out and ... may even be designed by pure calculators. (Guerreau 1981, 870)

In Guerreau's mind, the main issue for his discipline was thus not the division of labor itself but rather the "division of know-how" implied by the ways in which computers were used in the humanities. That idea of computer people being keen to create and maintain secrecy about their practice was nonetheless shared even by some computer scientists at the time (Backus 1980).

Guerreau then raised the second issue he has identified (material vs. intellectual), which originated from the discovery of new computer-assisted data analysis procedures. In the French academic world, the most famous procedure of this type at the time was correspondence analysis, which was invented by the aforementioned mathematician Benzécri and his team in the early 1960s at the University of Rennes (Benzécri 1982; Bonnet 2015; Escofier 1965; Martin 1997). Roughly described, correspondence analysis is a multi-dimensional analysis that reveals, in the form of a plot, the relative relationships between the rows and columns of a given dataset structured in a contingency table (Cibois 2000; Escofier 1965; Hjelbrekke 2018).

As other historians have already claimed elsewhere (Fogel 1975), Guerreau noticed that the reception of such complex mathematical procedures within the historical discipline was far from simple because of the historians' lack of statistical training (Lejeune 2022). That pitfall was even reinforced by a dispute that took place outside the historical discipline. In the early 1970s, correspondence analysis was indeed criticized by some of Benzécri's colleagues because of the neo-thomist and firmly realist epistemology he defended. The fashion in which he diffused his mathematical research, by refusing discussion with his contradictors and creating his own scientific events and publications, also provided fuel for his detractors (Cibois and Jambu 1981).

In his article, Guerreau deplored this context of distrust and its effect on the historical discipline. He argued that it limited the reception of correspondence

analysis in the social sciences and the benefit of another trend on how to use statistics in social science, strongly oriented toward the application of descriptive statistics. Facing that critical situation, he concluded as follows:

I have distinguished four groups of problems: 1. the tools of calculation; 2. the social organisation of calculations; 3. the forms and techniques of calculation; 4. the imposition of a methodological orientation by a dominant trend ... I propose here the combination of two material choices (point 1 and 3): the use of a programmable pocket calculator; the complementary use of correspondence analysis and classical methods; these choices are linked to two social positions (2 and 4): an expressly artisanal conception of social science research, a choice of method in favour of statistics linked to probabilities. (Guerreau 1981, 871)

Guerreau then justified more precisely his choice towards the programmable desk calculator by developing some of its advantages compared with mainframe computers:

Now a multitude of statistical analyses, which only a few years ago were mainly theoretical, can be carried out at home with speed and precision. Even small factorial analyses can be carried out perfectly: the use of a computer is no longer a prerequisite but only a possibility offered in the case of "heavy processing," a possibility which can become much more fruitful if one knows the details of the calculations and the multiplicity of the possible statistical analysis procedures himself. It is therefore not a question of opposing pocket calculators and computers, but of understanding the necessary complementarity of these two tools and of tending towards the disappearance of the magic practice of arrogant dissertation on indexes calculated according to a procedure of which one knows nothing. (Guerreau 1981, 871)

The two latter developments are of huge interest to understand the competition that took place at the time between programmable pocket calculators and mainframe computers.

First, it acknowledges that Guerreau, contrarily to Herlihy and Klapisch-Zuber, made a very informed choice, based on a precise representation of what computer-assisted methodology is in the humanities. His decision relied on the possibility of performing a clearly identified statistical procedure, correspondence analysis, as well as traditional statistical operations, "at home," with "speed and precision." For the purpose of doing so, Guerreau needed a calculation tool that could program a sequence of operations.

That strong choice towards correspondence analysis could be explained by several influences of his own academic world. The strongest was Guerreau's firm interest in structuralists' methodologies, and especially for the

computer-assisted methods developed by Bourdieu in the early 1970s (Bonnet 2015; Lejeune 2021). In the late 1970s, the use of correspondence analysis was also firmly established within the historical discipline, following the publications of Demonet, Prost and Klapisch-Zuber (Klapisch-Zuber and Demonet 1972; Prost 1974).

Second, Guerreau refused to work with a mainframe computer because of his will to carry out and to design the calculation by himself. By choosing so, he was in fact taking a stand against the paradigmatic model of history computing described in the previous section, where the division of labor between historians, computer scientists and computer center operators was mandatory. Two main arguments motivated his position. On the one hand, he was convinced that the historian has to know the statistical procedures applied to his data in order to better understand the results. On the other hand, he strongly refused any dependence on administrative and university power, e.g., computing centers, the CNRS, or other powerful institution such as the VIth section of the EPHE.

If that choice seems much more enlightened than that of Klapisch-Zuber and Herlihy, it must nevertheless be noted that Guerreau did not work in the same technological context than his older colleagues. In the mid 1970s, almost 10 years after the beginning of the *catasto* survey, the commercial offering for digital programmable calculators was already incredibly extensive. The commercial, professional and scientific literature designed to help customers buy their machine was also plethoric (Haigh and Ceruzzi 2021; Jouenne 2003).

With that context in mind, Guerreau's rationale with regard to his computing equipment was very different from that of Klapisch-Zuber and Herlihy, as he did not intend to produce a collective scholarly instrument such as the "edition for the computer" of the 1427 *catasto*. His orientation towards computing was in fact way more similar to the older and aforementioned works of the Oswleys, Curti or the Baillyns than it was from the ambitions of Busa, Tombeur, or Genicot.

As I will show in the next section, that difference in the goals attached to the use of computers was not the only one that shaped the disparities between these two local cultures of computing, as mainframe computer and programmable pocket calculators implied very different computing practices.

2. Computing Practices: Technical Details, Social Organization, and Cultural Transmission

In this section, I will discuss how the choice of a given computing instrument shaped the knowledge cultures of medievalists in the 1970s. I will first present the way in which I consider it appropriate to study a scientific culture, considering the fact that, in this specific context, new types of texts and intellectual technologies arose while pre-existent ones saw their functions transformed.

I will then focus on different elements of these computing cultures: the ways in which the datasets influenced and were influenced by the choice of computing devices; the practices of programming implied by the use of a Texas Instrument TI-58; and finally the data visualizations produced in both Guerreau's and Herlihy and Klapisch-Zuber's cases, keeping in mind the fact that Guerreau, contrarily to his two colleagues, did not print but hand-traced all his plots and maps.

2.1 Scientific Cultures in Question: Practices of Texts and Intellectual Technologies

A wide variety of approaches for studying scientific cultures have been developed in the past 50 years within the history of science literature. Nowadays, the issues of identifying, defining and studying a given scientific culture could even be considered one of the most dominant issues in the history of science. Paradigm, styles of thought, styles of reasoning or epistemic cultures—many of the latter concepts carry with them important corpora of debates and discussions. In a recent book on the subject, Chemla and Fox Keller (2017) addressed that diversity by calling for another approach to scientific cultures aimed at the development of “a history of culture making” rather than a history of close, immobile, and defining ones.

In my PhD dissertation, I have shown that, in order to approach a given digital humanities culture in the making, the historian can rely on the uses of the working documents produced (Lejeune 2021). For the case of the *catasto* survey, I have for example demonstrated that the constitution of that local culture relied on the development of specific knowledges, attached to the filling of the coding sheet designed by Herlihy and Klapisch-Zuber. That common knowledge then structured the existence of that group of scientists, where every scholar had to learn both what pieces of information from the archive material they should focus on and in which way they had to code them.

The input of the data onto this coding sheet thus relied on learning how to manipulate several types of texts: the declarations of the *catasto* themselves—with all the historical, palaeographical, and linguistic difficulties attached; a list

of all the districts of Tuscany—on which relied the division of labor between the different members of the team in charge of the input of data; and the coding manual, produced with the aim of helping each collaborator write the appropriate alphanumeric code for each piece of information found in the source (Lejeune 2021).

We can apply the same approach to the other part of the team, that is, the one in charge of the statistical analysis of the dataset. Klapisch-Zuber, Demonet, Herlihy and his computer scientists also needed to master the know-how related to the coding sheets. Yet, they also shared an array of texts that defined another facet of the local culture they were building: computer programs, called HHL or MARSTA—written in order to “give percentage,” to “summarize,” to rank, to draw “elegant graphs,” “scattergrams” or to calculate correlation coefficients, and the output documents printed from these programs—lists, tables, diagrams, and computer-generated maps (Lejeune 2021).

In order to understand the specificity of Guerreau’s computing practices, I propose here to follow a similar methodology. I will focus on the type of texts he produced and handled, rather than to study the manual and second-hand literature he relied on in order to learn how to program his TI-58. I will then rely on the working documents he published (Guerreau 1979a, 1979b, 1981) as well on an interview I conducted with him in 2017.

However, I must clarify that, among the numerous working documents he mentioned in his articles, only a few were published and thus remained accessible for study. Coding forms are one of these. They were provided with the purchase of the Texas Instrument TI-58 and consisted of a form designed to write the sequence of operations of a given program (see Figure 2). As we will see, they are of prime interest because they provide several hints on how Guerreau wrote, stored, and re-used a given program.

Another key material for my study is a corpus of published diagrams and maps. Their relations to the computing practices of Guerreau are, in my opinion, of higher importance as they rely on the pocket calculator programs but were not printed with a computer device. Guerreau drew these figures by hand on graph paper, before editing them with tracing paper and dry-transfer symbol sheets. This *modus operandi* appears as a very unique practice to be investigated in that context. It highlights how computing also implied numerous operations outside the computing machine at stake.

A brief comparison between these two corpora led to a first important methodological remark: the historian of computing has to take into account the materiality and structures of data in order to understand the specificity of a given computing practice. Are the data recorded onto punched card, onto magnetic tape, or onto paper? Are these data structured as a binary tree, as a

list, as a table, as a relational (or non-relational) database, etc.? I will start by analyzing how these datasets were used during computing activities.

2.2 Big Data and Small Data

The first major difference between my two case studies concerns the size of the dataset that the two group of actors considered to analyze. As I insisted in section 1.1, the *catasto* survey, conducted between 1966 and 1978, relied on the ambition to apply statistical procedures to an immense mass of information. In order to do so, the mainframe computer was considered an appropriate tool. Guerreau's choice toward the pocket calculator, which occurred in the mid 1970s, implied focusing on a rather smaller dataset, as he explained in a 1981 article:

The capacity of the calculator I used limited at 7 the number of columns in the tables so it was possible to use correspondence analysis, but the number of rows remained without limitation. (Guerreau 1981, 871)

That first element of disparity is of very high importance in order to understand how the choice of a given computing instrument shaped a given computing practice—without forgetting that each of these actors deliberately chose their instruments.

Contrarily to the *catasto* survey, Guerreau acknowledged that his will to apply correspondence analysis constrained the size of the dataset he could analyze, and more specifically its number of rows. If we imagine Klapisch-Zuber and Herlihy using the same computing device, the 60,000 declarations of the *catasto*, which could be considered as many rows, would not have been considered a theoretical issue contrarily to the 160 columns of the coding sheet, which represented 20 variables.

We also have to acknowledge, with Guerreau, that the possibility that the machine could deal with an infinite number of rows was just a “theoretical” assumption. For statistical processing, the multiplication of rows in a dataset implies a multiplication of elements in each of the calculations performed with the pocket calculator. For example, to calculate the household average wealth of Florentine taxpayers in 1427 with a TI-58, Herlihy and Klapisch-Zuber should have entered by hand the 60,000 values requested: an error-prone and time-consuming activity. The difficulties attached to the operation of a more complex procedure on a similar dataset are then left to the reader's imagination.

Nonetheless, these limitations imposed by the TI-58 were never considered of great importance by Guerreau, as he argued in his aforementioned article:

Let's then observe that, in medieval history, it is quite rare that one can draw from the documentation tables which have both more than 10

rows and more than 10 columns. As the columns and rows are strictly permutable, and that it is always advantageous to have fewer columns than rows, it is clear that a 7-column table can generally be obtained without any appreciable loss of information, and that consequently the medievalist is only moderately affected by this limitation. (Guerreau 1981, 871)

That argument is of huge interest for our subject because it highlights two other reasons for Guerreau's choice toward the TI-58. First, he noticed that medieval sources are not as rich as one can imagine, and that the TI-58 enabled him to deal with a lot of existing documentation. Then, he acknowledged the fact that the possible permutation of columns and rows, which is one of the specificities of the correspondence analysis, almost always enabled him to apply that procedure to his dataset.

Despite that apparent adequacy between the capacity of the computing device and the data available to a medievalist, I must insist on the existence of several other differences between the mainframe computers and pocket calculators. The most obvious one relied on the fact that the TI-58 could not handle alphabetical symbols. Because of that, Guerreau could not perform any operation on a dataset composed of nominal variables. For example, he would not have been able to study the family names of a given source, contrarily to what Klapisch-Zuber and Herlihy did for the case of the *catasto*.

That first limitation strongly affected the historical questions Guerreau considered with the help of his machine. Yet, it was not the most important discrepancy between these two types of computing practices. For the case of Guerreau, the dataset was structured as a table, composed of individuals in rows and numerical or categorical variables in columns—or *vice versa* if the number of variables was superior to seven and the number of individuals inferior to seven.

Herlihy and Klapisch-Zuber created a much more complex dataset, as they wanted to include both socio-economical and demographical information about the people living within the household. Thus, their dataset was structured with two tables, one for each type of information, where tables were related by the name of the householder. This database-like structure enabled operations to be performed on socio-economical data on the one hand and demographical data on the other, while offering the possibility of cross-analyzing both types.

That difference in the structure of the dataset reveals another dimension of the correspondence between small data and the choice of a pocket calculator. The latter machine was in fact designed exclusively for the purpose of calculation, and was not meant to perform any data-processing operations.

As the data were never input into digital storage, all of the data-processing operations were then done by hand by Guerreau.

For example, Guerreau managed to apply correspondence analysis to a dataset composed of 609 mendicant convents in 1981 (Guerreau 1981). In order to do so, he developed several strategies to make the computation lighter. He notably divided the whole population into smaller ones by defining seven chronological layers. The latter decision implied establishing a chronological list of all the convents and defining classes in a way in which every class was roughly composed of the same number of individuals. That rather complex procedure led Guerreau to sort, count and compare several different lists of mendicant orders, without any help from his machine.

The same analysis could be applied to other types of statistical operations, for example the calculus of percentages, means or coefficients. What it shows is that the use of a pocket calculator was adequate for small dataset not just because of the computing power of these machines, but also because of the fact that, in order to analyze a huge dataset, a given scholar needed to do a lot of time-consuming data-processing operations. It was then a prerequisite for one to be helped by other people. The case of the *catasto* survey once again provides a great illustration of the importance of these data-processing operations, as it took almost eight years for Klapisch-Zuber and Herlihy's team to edit the whole dataset onto magnetic tapes, despite the help of several collaborators.

These considerations about the correlation between the size and structure of the dataset and the computing device chosen by these scholars highlights two major elements. First, the time and pace of the computing practices at stake derived from the size of the dataset, which itself derived from the type of computing device chosen. Second, these two machines implied very different practices for data-processing operations. In the case of Klapisch-Zuber and Herlihy's project, the great bulk of the work was the edition of the data onto 80-column IBM punched cards. Their major challenge was to produce a single dataset that would be analyzed in many different ways with the help of the computer. Once that "edition for the computer" was ready, they could "program" it in many different ways, depending on their hypothesis, their ideas, and the consideration of the results provided by their statistical analysis.

In the case of Guerreau, all the operations that implied manipulation of data were more flexible, as they were done by hand, with the help of traditional intellectual technologies such as lists and tables. Yet, they were also more time-consuming as the dataset gained in size, which explains why he worked on a smaller dataset. As we will now see, these major differences derived from the specificity of the programming practices implied by the use of the two very different computing devices.

2.3 Programming a Correspondence Analysis on a Texas Instrument TI-58

In the past 10 years, historians of computing have studied various examples of what programming was in various contexts (Bullynck and De Mol 2010; Priestley 2011). These studies have shown how much programming practices depended on the computer equipment at stake and on the procedures that needed to be performed. They have also shown that many different types of intellectual technologies, e.g., diagrams, tables, and texts, were produced for the purpose of programming a computer (Bullynck and De Mol 2010).

To my knowledge, no literature has dealt with the practices of programming within the history of humanities computing. From my personal experience of the archive funds of humanities scholars using computers in Europe, this silence could be explained by the fact that documents related to programs and/or programming activities are very rare. Fortunately, the case of Guerreau's personal computing practices helps us in addressing this gap, as he was one of the rare historians, to my knowledge, who published working documents as well as articles dedicated to his programming practice (Guerreau 1979a, 1979b, 1981).

In a small text published in 1979 in the newsletter *Le Médiéviste et l'ordinateur*, Guerreau presented how to program a small procedure he called LSD. It was designed to convert the medieval monetary system into the decimal system. In the first section of his article, Guerreau briefly presented the operation of his TI-58 "programming mode":

Such a calculator has a set of memories, all addressable, which can store numerical data or instructions. If the LRN (learn) key is pressed, the calculator goes into "programming mode": all the keys pressed successively write a signal into the memory equivalent to the instruction they allow to execute directly in "calculation mode." There are several ways to execute a program. The simplest way is to use a "label": one or more keys are called by a simple letter (A, B, C, ...). If you place one of them at the beginning of a program, it will be sufficient, in "calculation mode," to press this key, and the calculator will execute the instructions that make up the program that starts with this label. (Guerreau 1979b)

Guerreau also attached to his article the Texas Instrument coding form presenting the 115 steps of LSD (see Figure 2).⁵

In that case, the program written by Guerreau took advantage of the simplicity of the programming language of these machines, the Specialize Machine Language (SML), characterized by the fact that every key of the keyboard

5 This decision relied on the particularity of the newsletter, which was designed to provide advice and basic computer knowledge to medievalists (Lejeune 2022).

TITLE TITRE				PAGE SEITE				OF VON				TI PROGRAMMABLE CODING FORM KODEFORM FEUILLE DE PROGRAMMATION			
PROGRAMMER PROGRAMMIER				DATE DATUM											
LSD				GUE RREAU											
LOC ADR	CODE CODE	KEY TASTE	COMMENTS BEMERKUNGEN	LOC ADR	CODE CODE	KEY TASTE	COMMENTS BEMERKUNGEN	LOC ADR	CODE CODE	KEY TASTE	COMMENTS BEMERKUNGEN	LOC ADR	CODE CODE	KEY TASTE	COMMENTS BEMERKUNGEN
0	16	LEE	introduction	10	16	LEE		1	00	+		1	00	+	
1	11	A	...aa/bb/cc	2	92	INT SBR		2	85	0		2	85	0	
2	42	STO	= L. S. d.	3	76	LEE		3	43	RCL		3	43	RCL	
3	00	00		4	10	E		4	01	1		4	01	1	
4	06	4		5	25	CLR		5	95	=		5	95	=	
5	22	INV		6	35	1/x		6	92	INT SBR		6	92	INT SBR	
6	22	log		7	43	RCL		7				7			
7	22	INV		8	00	0		8				8			
8	49	Pad		9	01	RIS		9				9			
9	00	0		10	76	LEE	introduction	10				10			
10	43	RCL		11	12	B.	en dernier	11				11			
11	00	0		12	42	STO		12				12			
12	53	INT		13	00	0		13				13			
13	22	INV		14	55	÷		14				14			
14	44	SUM		15	02	2		15				15			
15	00	0		16	04	4		16				16			
16	65	X		17	00	0		17				17			
17	02	2		18	95	=		18				18			
18	04	4		19	53	INT		19				19			
19	00	0		20	42	STO		20				20			
20	95	=		21	01	1		21				21			
21	42	STO		22	65	X		22				22			
22	01	01		23	02	2		23				23			
23	02	2		24	00	0		24				24			
24	00	0		25	45	=		25				25			
25	15	E		26	22	INV		26				26			
26	53	INT		27	44	SUM		27				27			
27	44	SUM		28	00	0		28				28			
28	00	0		29	04	4		29				29			
29	65	X		30	22	INV		30				30			
30	01	1		31	28	log		31				31			
31	02	2		32	49	Pad		32				32			
32	95	=		33	01	1		33				33			
33	44	SUM		34	43	RCL		34				34			
34	01	01		35	00	0		35				35			
35	01	1		36	55	÷		36				36			
36	02	2		37	01	1		37				37			
37	15	E		38	02	2		38				38			
38	44	SUM		39	95	=		39				39			
39	01	1		40	53	INT		40				40			
40	43	RCL		41	65	X		41				41			
41	01	1		42	01	1		42				42			
42	92	INT SBR	Résultat	43	00	0		43				43			
43	76	LEE	en dernier	44	00	0		44				44			
44	15	E		45	35	1/x		45				45			
45	32	X<2		46	44	SUM		46				46			
46	04	4		47	01	1		47				47			
47	00	0		48	65	X		48				48			
48	00	0		49	35	1/x		49				49			
49	49	Pad		50	04	4		50				50			
50	00	0		51	02	2		51				51			
51	43	RCL		52	94	+/-		52				52			
52	00	0		53	85	0		53				53			
53	72	X<2		54	43	Exc		54				54			

MERGED CODES
 KOMBINATIONEN CODES
 TOUCHES COMBINÉES
 62 [inv] 72 [inv] 83 [inv]
 63 [inv] 73 [inv] 84 [inv]
 64 [inv] 74 [inv] 92 [inv]

TEXAS INSTRUMENTS

[Fig. 2] Texas Instrument coding sheet of Guerreau's LSD program (source: Guerreau 1979b)

of the TI-58 was assigned with a given function. With such an interface, the scholar has to press a sequence of buttons in order to program, just as if he was performing each procedure step by step. Two main differences could be noticed if we compare that interface with an older mechanical calculator: the existence of a set of elaborate functions, such as logarithmic or trigonometric ones, and the possibility of storing programs in the internal memory of the machine.

However, the TI-58, contrarily to the TI-59, proposed no external device to store an existing program. This characteristic compelled Guerreau to record his program, step by step, at the beginning of each working session. In that context, the coding forms acted more as a storing device rather than as a medium that was meant to elaborate or communicate a given program—even if it became one with its publication in 1979.

Another example highlights how this apparent simplicity shaped the computing practices of Guerreau. In an article published in 1979, the French medievalist presented how to program a correspondence analysis on his programmable pocket calculator TI-58. In that case, if Guerreau did not reproduce the related coding form, the medievalist provided two important advices to the reader:

1. to reduce the duration of the work to a minimum by making the best use of the memory capacity of the machine.
 2. to check the accuracy of the results as frequently as possible so that only the minimum of calculations need be repeated in case of error.
- (Guerreau 1979a, 11)

Moreover, these two recommendations were attached to another difficulty, directly linked to a constraint imposed by the machine: the need to decompose the program into smaller segments because of the TI-58's limited internal memory, allowing the internal storage of no more than 480 sequence steps.

These practical advices are of great help to understand how the low internal memory of the TI-58 impacted Guerreau's computing practice. Contrarily to the workflow developed by the *catasto* team, Guerreau's practice of programming was in fact very close to that of a statistician using a mechanical calculator. First, as I have mentioned above, Guerreau needed, for each statistical procedure, to enter the value of each piece of information by hand. Second, the absence of a digital storage device for recording a program, combined with the lack of internal memory, implied that he had to re-enter each program segment step by step every time he wanted to perform a correspondence analysis, resulting in a way more time-consuming practice of computing.

These two elements (program segmentation and ephemeral storage) implied a very different pace of computing than that of Klapisch-Zuber and Herlihy, who relied on batch processing (see Haigh and Ceruzzi, 2021). A striking anecdote by Klapisch-Zuber emphasizes that difference. At the beginning of their study, Demonet and Klapisch-Zuber were at the computer center and started for the first time a program designed by Demonet. Because of an error in the writing of the program, the computer started to print an immense table. When Demonet realized that the printing would take hours, and thus make them

lose both precious and expensive computing time, he asked the operator to stop its execution.

Such an accident would have never happened to Guerreau, as he was always in control of the operations he was performing. However, these operations took him much more time than his two colleagues. In his 1979 article, Guerreau indicated that the computation of the eigenvalues of a seven-column matrix took him one hour, because of the need to perform 50 iterations of Jacobi rotations. The same operation, once programmed on a mainframe computer, took less than a minute. Moreover, this workflow was an error-prone process, and errors triggered by all the manual operations were one of the fundamental issues at stake for Guerreau, as he emphasized in his aforementioned 1979 article.

Considering these differences, one might wonder whether Guerreau's practices could be considered a "modern" programming practice. On the one hand, Guerreau used his computing device almost as a mechanical desk calculator, because of the lack of internal memory, resulting in manual and step-by-step programming that could be considered a simple "formalization" of older practices. On the other hand, Guerreau took advantage of the capacity of his TI-58 to automate computation in order to perform complex multivariate statistical procedures, or to easily replicate a pre-programmed sequence of operations.

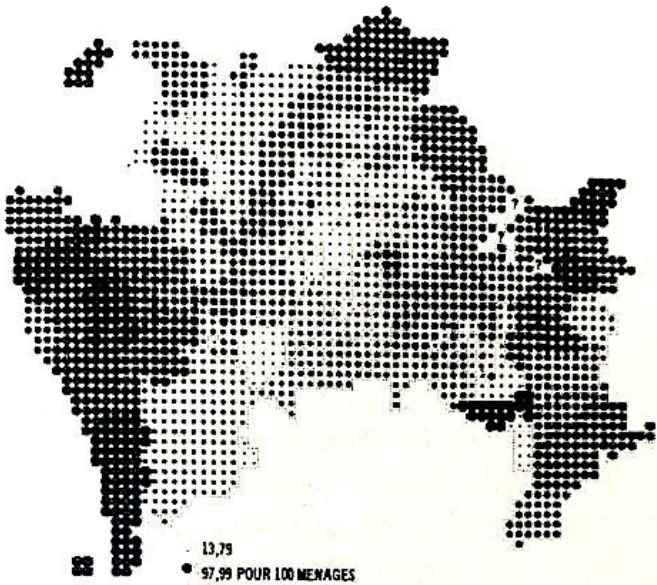
These two considerations are, in my opinion, two complementary characteristics of what computing is in the humanities. Following the approach of the computer scientist and French archaeologist Jean-Claude Gardin (1925–2013), programming appears essentially in the humanities as a "formalization" of older reasoning or, in other words, as an explication of "algorithmized" procedures (Gardin and Jaulin 1968). However, the capacity of the machine to automatically perform mathematical procedures, encapsulated in the machine, is also critical in order to define what programming is for Guerreau in this context.

Considering the latter, we can then describe Guerreau's practices as a minimalist form of programming, where the minimum amount of operations was stored and automatically operated by the machine. As I will show in the next section, the way in which he produced and edited the output documents reinforce that thesis.

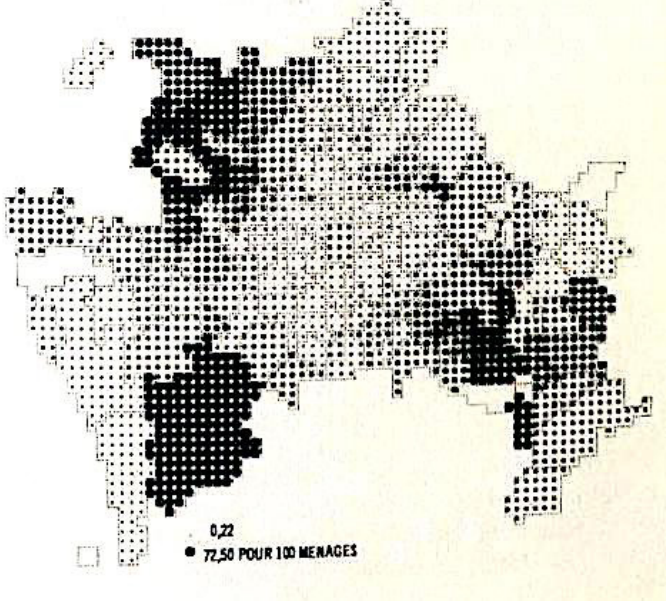
2.4 Printed or Handwritten Visualizations: Maps and Plots

Another dimension of computing practice is key to understanding the differences that existed between computing with a TI-58 and computing with a mainframe computer: the ways in which visualizations were produced

114 SANS PROFESSION



111 PROPRIÉTAIRES EXPLOITANTS



[Fig. 3] Maps produced by Bertin's laboratory for the *catasto* survey (source: Herlihy and Klapisch-Zuber 1978, 285)

in these local cultures of computing. As I have mentioned above, the main technological difference between the two contexts under study here is the presence (or not) of a printer.

In order to understand the importance of this device for the actors, we have to acknowledge that computer-generated visualizations are very different from those produced by other means, as they resulted from activities of computation. Thus, they need to be considered as results of analysis. In that perspective, they have a heuristic, or exploratory, value which is very different from that of illustrations.

A very striking example is provided by the maps produced during the *catasto* study by the team of cartographers led by Jacques Bertin in Paris. In the introduction to his 1967 treatise, *Sémiologie Graphique*, Bertin states that visualizations had undergone a fundamental change under the pressure of computer technology:

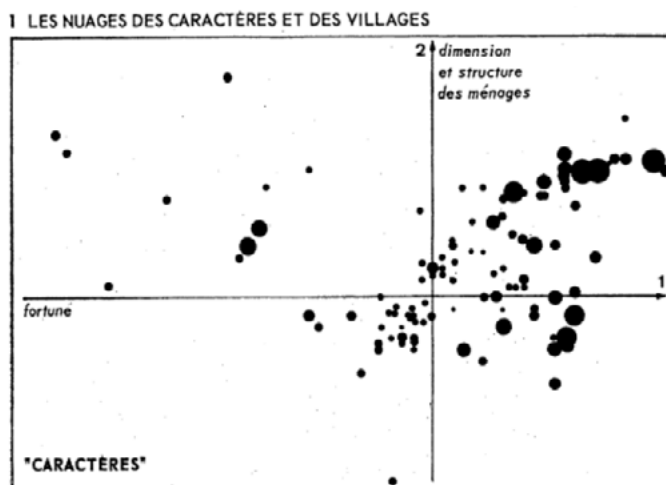
The graphic image has become manipulable through superimpositions, juxtapositions, transformations, permutations, allowing groupings and classifications, and has gone from being a *dead image*, an illustration, to a *living image*, a research tool accessible to all. (Bertin 1973 [1967], 8, italics in the original)

In Bertin's view, the function of graphic images no longer operates exclusively as a means of communication, but also as a heuristic instrument for processing information.

Two main characteristics underpin the value of the maps for our subject. The first concerns their modes of production. The maps produced for the *catasto* survey are, for the vast majority, the result of statistical processing of data provided by the computer "edition" of the source. They are printed automatically using a "dot printer" system, and the results took the form of the maps presented in Figure 3

Because of the automatic way in which they were produced, these maps have the same status as other output documents produced by a computer, such as a contingency table, a calculation of a relationship coefficient, or a plot of a correspondence analysis. In many cases, these maps even showed the medievalists unpredictable results. In a letter of August 12, 1971, Herlihy assesses the value of the latter in the following words: "mapping has nothing to do with geography, it is computer language ... a process which fits together all parts of a program."⁶

6 Letter by Herlihy, August 12, 1971, EHESS CRH CKZ 1–13. Fond Christiane Klapisch-Zuber, Humathèque Condorcet, Aubervilliers (France). See also (Lejeune, 2021, 118).



[Fig. 4] Correspondence analysis plot produced in the context of the *catasto* survey (source: Klapisch-Zuber and Demonet 1972, 895)

If maps are striking examples of the added values of printers in these contexts, another type of visualization produced in the context of the *catasto* survey was meant to provide unpredicted results: correspondence analysis plots (see Figure 4).

Just as other multivariate statistical procedures, correspondence analysis was meant to discover relations between rows and columns in a given dataset. From that perspective, the projection on the plot was meant to give the scientist a graphical representation of these relations on a selected two-dimension plane that provided a “good summary” of the total information.

Considering that function of the correspondence analysis plot, Demonet and Klapisch-Zuber relied on its printing by the computer in order to explore their dataset and to provide historical conclusions, just as they have done for the computer-generated maps. The case of Guerreau, who did not use a printer but traced his correspondence analysis plot by hand, then raised an array of epistemological questions that will need to be examined in more length.

Are there significant cognitive or epistemological differences between these two forms of practice? Or could we consider Guerreau's hand-tracing practices as equivalent to an automatic printing process? The latter questions are, in my opinion, of huge significance for our field of research, where automatization is considered the backbone of computing. However, my corpus as well as my methodology appear inadequate to answer these questions. Here is then an avenue of research on which I want to dedicate further analysis.

3. Conclusion

This article wishes to address two related issues: How does a researcher choose a computation tool from an array of available devices, and how does their equipment choice shape their practice of computing? In asking so, this article aims to provide fuel for two major contemporary debates in the history of computing. First, what should be considered a computer within our historical approach to computing? And to what extent should automatization and programming functions be considered crucial characteristics for defining modern computers? Second, what is the history of computing the history of? Is it concerned with the history of people “using” computers? And in that direction, what does “using a computer” historically mean?

I have shown that many different devices provided help for computing activities in the 1970s. Digital pocket calculators are, in that regard, of huge interest as they constitute a borderline case compared with mainframe computers. From the late 1960s, numerous models were marketed and provided very different functions to the purchaser: some were programmable and others were not; a few were able to perform more than the four classical arithmetic operations, and especially logarithmic and trigonometric functions; and others even came with a printer for diagrams and tables.

The comparison between the two case studies has revealed that different rationales were at stake when humanities scholars chose their equipment. On the one hand, computers were required for the purpose of storing and processing huge datasets. From that perspective, actors had the ambition to develop new types of scholarly instruments that could be easily reproduced and easily programmed. On the other hand, I have shown that another rationale was shared among the historical discipline, that is, the will to use computers in order to program statistical procedures, and especially new types of multivariate analysis operations.

These two rationales were related to two essential functions of modern computers. The first is related to the capacity of computers to store data and to perform data-processing operations on huge datasets. The second is once again related to the storage capacity of the computers, but is oriented toward the fact that some computing devices were able to store a sequence of instructions and its intermediary results.

The comparison between Herlihy's logic of choice and Guerreau's is in that regard of fundamental interest to our topic. If Herlihy and Klapisch-Zuber were interested in mainframe computers for the two aforementioned reasons (storing huge datasets and programming statistical procedures), Guerreau decided to purchase a pocket calculator only because of his will to program statistical procedures. The common rationale between these two groups of

actors thus provides elements in favor of the thesis that programming statistical procedures was one of the prominent reasons for historians to add a new scientific instrument to their toolbox in the 1970s.

However, I have shown in the second section of this article that the uses of these two very different computing devices implied very different computing practices. The analysis of Guerreau's practice of programming has highlighted the particularity of the uses of pocket calculators in that context. The main issue at stake here is the possibility offered by the machine to automatize certain tasks. For the case of Guerreau, I have shown that very few computing operations were in fact automatized because of the small internal memory of his TI-58.

That discrepancy implied that a lot of the computing operations done by Guerreau were in fact manually performed operations: data processing, entering data, storing programs for long time periods, or tracing plots and maps. All these manual operations affected the computing practice of Guerreau in many ways, because of both their time-consuming and error-prone nature. Nonetheless, it drastically limited the black box effect of the use of a mainframe computer for a humanities scholar. In doing so, Guerreau had to master the statistical procedures he wished to apply as well as to be familiar with programming. He then became a recognized figure in statistics and computer science within medieval history, and even trained younger historians who are now in charge of important digital humanities curricula in prestigious institutions in France.

That example of an original computing practice thus helps us understand that the passage from "manual" computing to automatic computing was not as brutal as one can imagine. Computing practices should then rather be analyzed in a continuum from manual computing to fully automatic data processing and computing. From that perspective, I have proposed in this paper a methodology that focuses on intellectual technologies and manual operations in order to define a given computing culture. I think that this approach could be of great help in order to insist on the diversity of digital scientific practices.

I thank Alain Guerreau, Christiane Klapisch-Zuber, and Michel Demonet, who accepted to be interviewed. I also thank Nicolas Veyssset and Béatrice Marin, who helped me access the archival collections of the CRH in Paris. I am also grateful to the people and institutions who helped me in the study of these materials: my two PhD supervisors, Karine Chemla, Shirley Carter-Thomas, and the entire SPHERE laboratory team (CNRS, Université Paris-Cité). Finally, I warmly thank the two editors of the book, Arianna Borrelli and Helena Durnova, and every contributor who provided their valuable comments on earlier versions of this article.

References

- Adams, Margaret. 1995. "Punch Card Records: Precursors of Electronic Records." *The American Archivist* 58, no. 2: 182–201. <https://doi.org/10.17723/aarc.58.2.d61078182725616j>.
- Aydelotte, William O. 1954. "A Statistical Analysis of the Parliament of 1841: Some Problems of Method." *Historical Research* 27, no. 76: 141–55. <https://doi.org/10.1111/j.1468-2281.1954.tb01131.x>.
- Backus, John. 1980. "Programming in America in the 1950s—Some Personal Impressions." In *A History of Computing in the Twentieth Century*, edited by Nicholas Metropolis, Jack Howlett, and Gian Carlo Rota, 125–35. Orlando: Academic Press.
- Bailyn, Bernard, and Lotte Bailyn. 1959. *Massachusetts Shipping, 1697–1714*. Cambridge, MA: Belknap Press of Harvard University Press.
- Béaur, Gérard, and Béatrice Marin. 2011. "La Statistique Générale de la France – Présentation." *L'Atelier du Centre de recherches historiques* March. <https://doi.org/10.4000/acrh.2891>
- Benzécri, Jean-Paul. 1982. *Histoire et préhistoire de l'analyse des données*. Paris: Dunod.
- Bertin, Jacques. 1973 [1967]. *Sémiologie graphique : Les diagrammes, les réseaux, les cartes*. Second Edition. Paris: Gauthier-Villars.
- . 1969. "Graphique et mathématique: Généralisation du traitement graphique de l'information." *Annales* 24, no. 1: 70–101. <https://doi.org/10.3406/ahess.1969.422034>.
- Bonnet, Philippe. 2015. "Pour une histoire sociale de l'analyse des données." In *La méthodologie de Pierre Bourdieu en action*, edited by Frédéric Lebaron and Brigitte Le Roux, 21–42. Paris: Dunod.
- Bullyncx, Maarten, and Liesbeth De Mol. 2010. "Setting-up Early Computer Programs: D. H. Lehmer's ENIAC Computation." *Archive for Mathematical Logic* 49: 123.
- Burke, Peter. 1990. *The French Historical Revolution: The "Annales" School, 1929–1989*. Cambridge and Stanford: Polity Press and Stanford University Press.
- Burton, Dolores M. 1981. "Automated Concordances and Word Indexes: The Early Sixties and the Early Centers." *Computers and the Humanities* 15, no. 2: 83–100. <https://doi.org/10.1007/BF02404202>.
- Burnard, Lou. 2012. "Du literary and linguistic computing aux digital humanities: retour sur 40 ans de relations entre sciences humaines et informatique." In *Read/Write Book 2: Une introduction aux humanités numériques*, edited by Pierre Mounier, 45–58. Marseille: OpenEdition Press.
- Campbell-Kelly, Martin. 2003. "Punched card machinery." In *Encyclopedia of Computer Science*, 1489–91. Chichester: John Wiley & Sons Ltd.
- Charbonneau, Hubert. 1995. "Louis Henry et la démographie historique." *Population* 50, no. 6: 1663–71. <https://doi.org/10.2307/1534477>.
- Chemla, Karine, and Evelyn Fox Keller, eds. 2017. *Cultures Without Culturalism: The Making of Scientific Knowledge*. Durham, NC: Duke University Press.
- Cibois, Philippe. 2000. *L'analyse factorielle: analyse en composantes principales et analyse des correspondances*. Paris: Presses Universitaires de France.
- Cibois, Philippe, and M. Jambu. 1981. "Analyse des données et sociologie." Edited by L. Lebart, A. Morineau, N. Tabard, M.-O. Lebeaux, J.-P. Fénelon, J.-P. Benzécri, F. Benzécri, J.-M. Bouroche, and G. Saporta. *L'Année sociologique (1940/1948–)* 31: 333–48.
- Connes, Janine, and Françoise Perriquet. 2022. De l'IBM 360/75 au superordinateur Jean Zay. 50 ans d'informatique au centre de calcul du CNRS d'Orsay. Les Ulis: EDP Sciences. <https://stm.cairn.info/de-l-ibm-360-75-au-superordinateur-jean-zay--9782987525776>
- Créhangé, Marion, and Lucie Fossier. 1970. "Essai d'exploitation sur ordinateur des sources diplomatiques médiévales." *Annales* 25, no. 1: 249–84. <https://doi.org/10.3406/ahess.1970.422211>.
- Crymble, Adam. 2021. *Technology and the Historian. Transformations in the Digital Age*. Urbana: University of Illinois Press.

- Curti, Merle Eugene, Robert R. Daniel, Shaw Livermore, and Joseph Van Hise. 1959. *The Making of an American Community: A Case Study of Democracy in a Frontier Country*. Stanford: Stanford University Press.
- Daumard, Adeline, and François Furet. 1959. "Méthodes de l'histoire sociale: les archives notariales et la mécanographie." *Annales* 14, no. 4: 676–93. <https://doi.org/10.3406/ahess.1959.2865>.
- de Tollaenaere, Félicien. 1977. "Les chartes néerlandaises en langue vulgaire." *Publications de l'École Française de Rome* 31, no. 1: 105–6.
- Duval, Julien. 2013. "L'analyse des correspondances et la construction des champs." *Actes de la recherche en sciences sociales* 200, no. 5: 110–23.
- Escofier, Brigitte. 1965. "L'analyse factorielle des correspondances." PhD Dissertation. Université de Rennes.
- Fogel, Robert William. 1975. "The Limits of Quantitative Methods in History." *The American Historical Review* 80, no. 2: 329–50. <https://doi.org/10.2307/1850498>.
- Fossier, Lucie. 1978. "Informatique et histoire médiévale à l'Institut de Recherche et d'Histoire de Textes." *Computers and the Humanities* 12, no. 11: 109–12. <https://doi.org/10.1007/BF02392923>.
- Furet, François. 1971. "Histoire quantitative et construction du fait historique." *Annales* 26, no. 11: 63–75. <https://doi.org/10.3406/ahess.1971.422459>.
- Gardin, J. C., and B. Jaulin. 1968. *Calcul et formalisation dans les sciences de l'homme*. Paris: Editions du Centre National de la Recherche Scientifique.
- Genicot, Léopold. 1974. "Index Scriptorum Operumque Latino-Belgicorum Medii Aevi. Nouveau répertoire des œuvres médiolatines belges publié sous la direction de L. Genicot et P. Tombeur. Première partie: VIIe-Xe siècles par A. Stainier (= Académie royale de Belgique. Comité national du dictionnaire du latin médiéval), Bruxelles, 1973." *Bulletins de l'Académie Royale de Belgique* 60, no. 1: 29–30.
- Grossetti, Michel, and Pierre-Eric Mounier-Kuhn. 1995. "Les débuts de l'informatique dans les universités." *Revue française de sociologie* XXXVI, no. 22: 295–324. <https://doi.org/10.2307/3322250>.
- Guerreau, Alain. 1979a. "Analyse factorielle au moyen d'une calculatrice de poche programmable." *Le médiéviste et l'ordinateur* 2, no. 1: 10–12. <https://doi.org/10.3406/medio.1979.905>.
- . 1979b. "Calculs directs, sans approximation, en livres-sous-deniers." *Le médiéviste et l'ordinateur* 2, no. 1: 13–15. <https://doi.org/10.3406/medio.1979.906>.
- . 1981. "Analyse factorielle et analyses statistiques classiques: le cas des ordres mendiants dans la France médiévale." *Annales* 36, no. 5: 869–912. <https://doi.org/10.3406/ahess.1981.282792>.
- . 1982. "Analyse statistique des finances municipales de Dijon au XVe siècle; observations de méthode sur l'analyse factorielle et les procédés classiques." *Bibliothèque de l'École des chartes* 140, no. 1: 5–34. <https://doi.org/10.3406/bec.1982.450254>.
- Haigh, Thomas, ed. 2019. *Exploring the Early Digital*. Cham: Springer.
- Haigh, Thomas, and Paul E. Ceruzzi. 2021. *A New History of Modern Computing*. Cambridge MA: MIT Press.
- Henry, Louis. 1972. "Simulation d'une reconstitution de familles par ordinateur." *Annales de Démographie Historique* 1972, no. 1: 303–309. <https://doi.org/10.3406/adh.1972.2130>.
- Herlihy, David, and Christiane Klapisch-Zuber. 1978. *Les Toscans et leurs familles: Une étude du catasto florentin de 1427*. Paris: Fondation Nationale des Sciences Politiques.
- Hjellbrekke, John. 2018. *Multiple Correspondence Analysis For The Social Sciences*. London: Routledge.
- Hockey, Susan. 2004. "The History of Humanities Computing." In *A Companion to Digital Humanities*, edited by Susan Schriebman, Ray Siemens, and John Unsworth, 1–19. New York: John Wiley & Sons.
- Jones, Steven Edward. 2016. *Roberto Busa, S. J., and the Emergence of Humanities Computing: The Priest and the Punched Cards*. London: Routledge.

- Jouenne, Noël. 2003. "Les calculatrices de poche." In *Actes du congrès d'histoire des sciences et des techniques*, May 2003, Poitiers, France ; 57–63. SFHST. Accessed November 2024. <https://halshs.archives-ouvertes.fr/halshs-00004544>.
- Klapisch-Zuber, Christiane. 1969. "Fiscalité et démographie en Toscane (1427–1430)." *Annales* 24, no. 6: 1313–37. <https://doi.org/10.3406/ahess.1969.422169>.
- Klapisch-Zuber, Christiane, and Michel Demonet. 1972. "A uno pane e uno vino: la famiglia rurale toscane au début du XVe siècle." *Annales* 27, no. 4: 873–901. <https://doi.org/10.3406/ahess.1972.422571>.
- Lejeune, Edgar. 2021. "Médiévistes et ordinateurs. Organisations collectives, pratiques des sources et conséquences historiographiques (1966–1990)." PhD Dissertation, Université de Paris Cité. HAL-SHS. Accessed November 2024. <https://halshs.archives-ouvertes.fr/tel-03598652>.
- . 2022. "Objectifs et stratégies de publication d'un bulletin de liaison : Le Médiéviste et l'ordinateur (1979–1989)." *Humanités numériques* 6. <https://doi.org/10.4000/revuehn.3087>.
- . 2024. "'Open' or 'Close' Research Instruments? Conflicting Rationales in the Organization of Early Digital Medieval History in Europe (1960–1990)." *Digital Humanities Quarterly* 18, no. 2. <https://www.digitalhumanities.org/dhq/vol/18/2/000741/000741.html>.
- Loiseau, Sylvain, and Jacqueline Léon, eds. 2016. *History of Quantitative Linguistics in France*. Lüdenscheid: RAM-Verlag.
- Mahoney, Michael S. 1988. "The History of Computing in the History of Technology." *IEEE Annals of the History of Computing* 10, no. 2: 113–25. <https://doi.org/10.1109/MAHC.1988.10011>.
- Martin, Olivier. 1997. "Aux origines des idées factorielles [Des théories aux méthodes statistiques]." *Histoire & Mesure* 12, no. 3: 197–249. <https://doi.org/10.3406/hism.1997.1544>.
- Mercourroff, Wladimir. 2015. "Témoignage : l'informatique dans l'enseignement et au CNRS au début des années soixante-dix." *Histoire de la recherche contemporaine*. Tome IV, no. 2: 180–85. <https://doi.org/10.4000/hrc.1154>.
- Netter, Marie-Laurence, and Christiane Klapisch-Zuber. 2005. "Cinquante ans d'histoire au Centre de Recherches Historiques." *Les Cahiers du Centre de Recherches Historiques. Archives* 36. <https://doi.org/10.4000/ccrh.3048>.
- Owsley, Frank Lawrence, and Harriet Owsley. 1949. *Plain Folk of the Old South*. Baton Rouge: Louisiana State University Press.
- Palsky, Gilles, and Marie-Claire Robic. 2000. "Aux sources de la sémiologie graphique." *Cybergeog: European Journal of Geography* Document 147. <https://doi.org/10.4000/cybergeog.554>.
- Plutniak, Sébastien. 2017. "L'opération archéologique. Sociologie historique d'une discipline aux prises avec l'automatique et les mathématiques. France, Espagne, Italie, 2e moitié du XXe siècle." PhD dissertation, Paris, EHESS. Accessed November 2024. <http://www.theses.fr/2017EHES0045>.
- Priestley, Mark. 2011. *A Science of Operations*. London: Springer.
- Prost, Antoine. 1974. *Vocabulaire des proclamations électorales de 1881, 1885 et 1889*. Paris: Presses Universitaires de France.
- Scalfati, Silio Pietro Paolo. 1977. "Problèmes de méthodes, à propos des actes notariés de Pise." *Publications de l'École Française de Rome* 31, no. 1: 251–52.
- Sula, Chris Alen, and Heather V. Hill. 2019. "The Early History of Digital Humanities: An Analysis of Computers and the Humanities (1966–2004) and Literary and Linguistic Computing (1986–2004)." *Digital Scholarship in the Humanities* 34 (Supplement 1): 190–206. <https://doi.org/10.1093/llc/fqz072>.
- Tatarchenko, Ksenia. 2019. "The Man with a Micro-Calculator": Digital Modernity and Late Soviet Computing Practices." In *Exploring the Early Digital*, edited by Thomas Haigh, 179–200. Cham: Springer.
- Tombeur, Paul. 1970. "Le Centre de Traitement Électronique des Documents, au service des disciplines ecclésiastiques." *Revue Théologique de Louvain* 1, no. 4: 433–43. <https://doi.org/10.3406/thlou.1970.1057>.
- Wenin, Christian. 1972. "L'informatique au service de la philosophie. Réalisation et projets." *Revue Philosophique de Louvain* 70, no. 6: 177–211. <https://doi.org/10.3406/phlou.1972.5663>.

Wisbey, Roy Albert. 1963. "The Analysis of Middle High German Texts by Computer—Some Lexicographical Aspects." *Transactions of the Philological Society* 62, no. 1: 28–48. <https://doi.org/10.1111/j.1467-968X.1963.tb00999.x>.

Yates, Frank. 1966. "Computers, the Second Revolution in Statistics." *Biometrics* 22, no. 2: 233–51.

EKISTICS

ARCHITECTURE

DATA

CYBERNETICS

ENVIRONMENT

Computing Practices, Data-Based Design, and Knowledge Cultures During the Post-War Period

Nathalie Bredella

Looking at the history of data-based design, this paper traces computational approaches developed in the post-war period in the context of ekistics, the science of human settlements founded by the architect and urban planner Constantinos Doxiadis. Supported by governments and humanitarian institutions, ekistics explored the analysis of settlements with a view to systems theory. Focusing on ekistics planning approaches, I am interested in the technical assemblages and cybernetically inspired practices that resulted in spatial protocols, and in the ways in which they functioned at the nexus of governance mechanisms (architecture, politics, and the environment), while operating at various scales (the tool, the city, and the planet).

In 1969, ekistics—the science of human settlements—founded by the Greek architect and urban planner Constantinos Doxiadis, dedicated an issue of its journal *Ekistics* to computers and their role in architecture and urbanism. The editorial that accompanied the journal's "Computers in the Service of Ekistics" issue captured the fears and hopes associated with the advent of the computer in architectural and urban practice.¹ Authors drew on cybernetic models to reflect on how different disciplines relate to and impact planning. The cybernetically inspired practices were also of interest to post-war governments looking to harness insights into the structure of communities in order to impact life through planning. While the computer was presented as a tool that could encourage novel approaches to planning and promote the optimistic notion of "mastering the big picture," the authors also weighed up its limitations. Researchers noted that databases were only as good as the data collected in the first place (Derry et al. 1969) and that conclusions drawn from statistical analysis had to be distinguished from more sophisticated human reasoning (Mead 1969). Although simulations could project a wide range of scenarios, these models were ultimately based on, and would perpetuate, past approaches (Steinitz and Rogers 1969). Notwithstanding the critical assessments of human-machine interaction, data-based techniques were presented as tools that would allow architects to develop expertise in the field and expand their role in advising policymakers. In particular, calculations related to the performance of the computer, such as the processing of large amounts of data and the enhancement of analytical processes, would provide knowledge on settlements and situate their infrastructures within environmental, economic, and social conditions, using media such as early computer games and mapping techniques (Taylor and Carter 1969), while case studies would capture, for example, the interdependencies between local economies and marine ecology (Isard et al. 1969).

The themes explored by the *Ekistics* journal were also the subjects of various architectural endeavors dealing with the interdependencies of architecture and computer technology at the time. The *Architecture and the Computer* conference held in Boston in 1964 focused on the various levels at which computers were being introduced into architectural practice, with students, architectural offices, and software companies taking part. Among the participants was student Emilio Ambasz from Princeton University, the architectural firm Doxiadis Associates (DA),² and the software company IBM, which

1 The editorial emphasized that the computer would be "kept in place," acting as a "mechanical slave" while "performing all the repetitive calculations and routine dog work," thus allowing "man himself the time to relate these facts to the development of society" ([Editorial] 1969, 1–3).

2 In the following, Constantinos Doxiadis is referred to by his surname and the firm is denoted by the abbreviation DA.

supported the conference with equipment and funding (Kotsioris 2015).³ Methods of collecting, processing, and visualizing data, as well as questions of statistics and creativity, were part of the broad framework within which computing in architecture was discussed. As in the *Ekistics* journal, presenters expressed a range of opinions, with some perceiving the computer as a slave or clerk and others pointing to its creative capacity; for example, Marvin Minski from MIT predicted that future machines would excel in creativity and intelligence (Minski 1964, 45). As advances in software and hardware were discussed by architects and information technology (IT) firms, governments and private foundations supported the expansion of computer networks and infrastructure through funding.

The Harvard Graduate School of Design (GSD), with a grant from the Ford Foundation, established the Laboratory for Computer Graphics and Spatial Analysis in 1965. Dedicated to the development of computer mapping technologies, the research focused on the development of computer programs that would relate statistical data to spatial distributions (Kotsioris 2022). Landscape planner Carl Steinitz, who together with Peter Rogers presented simulation studies in the above-mentioned issue of *Ekistics*, was involved in the lab, where he explored the graphics of computer maps and their impact on urban practice (McMahon 2013, 437). Through the adaptation of spatial mapping systems such as SYMAP (Synagraphic Mapping System)—originally launched at Northwestern University—the GSD engaged in research on Graphical Information Systems (GIS).⁴ DA closely followed these developments and expressed interest in seminars on SYMAP to expand their knowledge of mapping techniques.⁵

In an age of technophilia, methods of data retrieval, visualization, and analysis were also key to post-war reconstruction programs, with so-called development aid programs promoting global economic and technological advances (Eberstadt 1997). Investment in development policy by the Eastern and Western blocs was thus part of a strategy to gain political and economic influence over the newly independent countries of Asia and Africa, with architects increasingly acting as consultants to governments and policy-makers (De Dominicis 2019, 193).

- 3 The proceedings of the conference were published with a grant from the Graham Foundation (Boston Architectural Center 1964).
- 4 Steinitz taught a design studio at GSD in which he explored the potential of SYMAP. On Steinitz's work at the lab, see McMahon (2013, 437–48); Bailey, Bredella and Steinitz (2025, 79–97).
- 5 In a note exchanged between the US and Athens offices of DA the latter (E. Fthenakis, Director of Education) expressed interest in a post-conference course on the Computer Mapping Program sponsored by Northwestern University in New Jersey on 20 and 21 August 1964, and also asked Howard Fisher from the Laboratory of Computer Graphics at GSD for relevant material on mapping techniques (Fthenakis 1964).

Large-scale projects commissioned by governments and funded by the Ford and Rockefeller Foundations also allowed DA to conduct research, develop design tools and explore data-based methods, while promoting interdisciplinary studies. As governments were arguing for modernization and economic growth, they turned to planning to develop and implement globally oriented strategies. As Ellen Shoshkes and Sy Adler (2009) show, Doxiadis's technocratic logic promised verifiable observations and allowed him to extend his expertise and authority. In this way, the ekistics approach to planning fit well with US and United Nations (UN) programmes promoting modernist planning schemes as a means of spreading freedom, peace, and progress in times of decolonization.

In what follows, I address the knowledge practices and scales on which ekistics operated, asking: How did the embrace of computer technology unfold in the context of ekistics, and what impact did computer firms have on the ways in which the computer became part of architectural planning and vice versa? How did data-based planning affect the understanding of architecture and its relationship to, and governing of, the environment? Finally, how have computing technologies been situated in the social and political realms? By focusing on the scale of the tool, the city, and the planet, I show how ekistics' engagement with computation was entangled both with post-war politics and with the interests of computer firms to spread computer technologies, and how the technological thinking it promoted affected architectural culture. I begin by outlining in slightly more detail ekistics' embrace of the computer before underscoring what was at stake for architecture and planning.

1. Ekistics and the Computer

Doxiadis began to formulate ideas about ekistics in the 1940s, before establishing DA, a consulting firm of engineers, architects, and planners, in 1953. By then he was a leading figure in Greece's reconstruction programs, having held posts in the Ministry of Reconstruction and Housing after the war, while also heading the Greek delegation to the UN International Conference on Housing, Planning and Reconstruction (1947) and the Greek-Italian Conference on War Compensation (1949–50). Shortly before the war, after completing his doctorate at the University of Charlottenburg in Berlin, he had served as chief town planner for the Greater Athens area. During the occupation, as leader of the national resistance group Hephaestus, he published the underground magazine *Regional Planning, Town Planning, Architecture*, in which, working with several underground movements, he collected and visualized data on the destruction of Greece by the German occupying forces (Biographical note n.d.). The group used the pictorial statistical display system known as Isotype (International System of Typographic Picture Education), as well as photographs, to map the Nazis' war crimes. At

the end of the war, the tables and diagrams documenting the war's devastation were exhibited in the *Survival of the Greek People* exhibition, first in Greece and then Paris, London, and San Francisco, where the material was displayed at the founding conference of the UN Charter (Kyrtis 2006, 347–48; Tsiambaos 2012, 51). The visualizations on display conveyed the destruction of the country. They also served as a reference point in the Greek nation's efforts to obtain financial aid from the Western Allies (Jansen 2009, 233).

As Doxiadis drew up plans for reconstruction, the study of past settlements was central to informing future schemes. His analysis of settlements was intended to allow knowledge of the past, specifically knowledge stored in buildings, to be harnessed for future projects by reclaiming qualities and applying them to settlements facing the contemporary challenges of population growth and ecological crises. When studying historic sites, the office introduced classification systems to document information about buildings, infrastructure, and their relationships at different scales of planning (Doxiadis 1972).⁶ This systematic approach, in turn, led to the identification of settlement patterns and was intended to increase understanding both of the specific cultural contexts in which such patterns arose and how they were transmitted across time and cultures. The social, geographic, and cultural insights gained from this research could then be applied when calculating solutions for urban planning. Ekistics' planning methods were thus based on an understanding of past and present settlements as sites of data storage and transmission.

The fact that Doxiadis was not only a practicing architect but also founded research and educational institutions was another hallmark of his data-driven approach. In addition to founding DA, as part of his effort to promote the study of technology and planning he also founded the Athens Technological Organisation and, in 1963, the Athens Center of Ekistics (ACE). Both institutions were dedicated to research and education. The founding of the DA Computer Center in 1969 was marked by a belief that technology, by enabling interdisciplinary collaboration and promising economic growth, was a means of meeting the challenges of the time, although where precisely the computer would be applied in the field of architecture was still unclear. Arguably, the rhetoric of progress associated with the computer gave Doxiadis the authority to address the political and economic challenges of the time and their relationship to planning, which in turn led to commissions from governments and international aid organizations as well as industry. In addition to being integral to the research and practice of ekistics, the computer center also formed an

6 In his dissertation on architectural space in ancient Greece (completed in 1937 at Charlottenburg University in Berlin), Doxiadis formulated his ideas on the organization of space and the application of scientific rules for its analysis and planning, revealing the geometric principles and mathematical correlations used in the design of public spaces and buildings (Doxiadis 1972). On Doxiadis's methodological approach to the study of the ancient Greek built environment see, Zarmakoupi (2015).

important part of the country's early computational infrastructure, providing digital technology services to Greece's political, industrial, and public sectors (Drymiotis 2007).

At ACE, students were encouraged to study a variety of subjects, including sociology, economics, demography, and transportation. Most important, the curriculum focused on the role computing could play in design (Theodosis 2015, 189). Seminars such as "Mathematics and Ekistics" furthered a computational-mathematical approach to architecture and urbanism and were taught alongside cybernetics, network analysis, and game theory (Doxiadis 1974). These themes are indicative of how the computer entered the field of planning through education. John Taylor and K.R. Carter published an essay on the role games and simulations could play for architectural education in the "Computers in the Service of Ekistics" issue of *Ekistics*. Presenting various game types, they explained how each allowed users or students to interact with and understand the dynamics of the urban fabric (Taylor and Carter 1969). Steinitz and Rogers featured studio work in the same *Ekistics* issue, reflecting on the possibilities and drawbacks of using simulation models in education to work on housing location, open space development, and spatial analysis (Steinitz and Rogers 1969, 60).

In Athens, computer technology was also involved in bringing research and education into line with the practical side of ekistics, as surveys and preliminary studies carried out by the ACE were passed on to DA (Theodosis 2015, 198). The links Doxiadis drew among institutions and between theory and practice, as well as his embrace of computer technology, not only underlined the interdisciplinary side of ekistics but also addressed the implications of media technologies for architecture and planning more generally.

Discussions about the transformations and challenges wrought by various media were among the subjects discussed at the Delos Symposia, annual events that Doxiadis, together with Jacqueline Tyrwhitt, organized from 1963 to 1975. The symposium brought together an interdisciplinary group of academics, politicians, and economists who spent a week by the Aegean Sea exploring not only how technology affects planning but also how people interact with and perceive the environment (Wigley 2001). Marshall McLuhan participated in the symposia and was joined by R. Buckminster Fuller in emphasizing the need for architects and planners to engage with electronic technologies to understand their spatial logic (Fuller 1965, 97). Data collection, visualization, and communication, as well as information networks were conceived as the media within which architecture operated, with infrastructures playing a central role, as they not only organized but also sustained the city. Architects' focus on organizational structures was often underpinned by a biological metaphor—the city as a living organism—with urban networks presented in analogy to the nervous system of a body. These relations proved

useful in conceptualizing the role of technological systems such as roads, water supplies, computers, and so on in the sustaining of settlements at both micro and macro scales (Doxiadis 1966; Wigley 2001, 87).

How the world's resources could be managed to ensure a more equitable distribution of human settlements was a pressing issue given the growing population and finite resources of the time. Fuller, together with John McHale, focused on networks and simulation games to produce dynamic data sets (Fuller 1969; McHale 1962), while Jay Forrester and his systems dynamics team at MIT explored early computational techniques to model and simulate urban and world dynamics for the Club of Rome.⁷ However, Doxiadis, a member of the Club of Rome, was still critical of the warnings expressed in *The Limits to Growth* (Meadows 1972), believing that technological advances could provide solutions to prevent the planetary collapse predicted by the report (Theodosios 2015, 346–437).

Doxiadis's embrace of technology was reflected in his collaborations with industry too. The Doxiadis Associates Computer Center (UNIVAC-DACC) collaborated with US computer companies and represented the Sperry Rand Corporation in Greece (Kyrtis 2006, 455). This partnership allowed the computer company to explore new markets, aligning it more closely with architecture. As ekistics integrated computers into its practice, Sperry Rand emphasized the potential of the computer for planning.⁸ Its press release "Cities of Tomorrow Planned on High-Speed Computers" captures the installation of high-speed Sperry Rand UNIVAC computers—initially a UNIVAC 1004 System and later a UNIVAC 1107—at DA and their potential for addressing the challenges of expanding cities (Maynard 1968). The release alludes to the planning schemes Doxiadis was realizing in the United States, France, and South America. Computers, the company reasoned, were well suited to manage complex sets of data on urban environments, as they could organize urban development by managing data on populations, including their economic, social, technical, and cultural factors.

The computer was promoted as a mapping tool that could create an "inventory" of a place—specifically, its transportation networks and utility services—based on data mined from the physical environment. Mathematical models in turn would allow the future development of a settlement to be simulated, with the characteristics and composition of a neighborhood considered alongside the movement of residents and economic infrastructures. The press release concluded that, in addition to generating a variety of assumptions, computers would systematically encourage interdisciplinary

7 On Doxiadis's role in the Club of Rome, see Scott 2016, 217, and also Doxiadis and Club of Rome (1970).

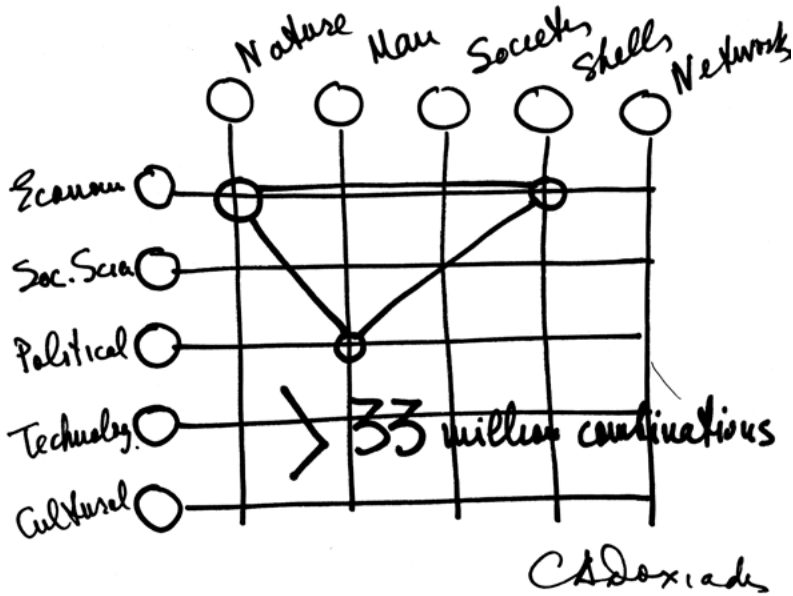
8 On computational firms' efforts to explore markets in various disciplines, see the essay by Arianna Borrelli in this volume.

thinking (Maynard 1968). The methods developed by ekistics were well suited to running simulations based on the calculations and visualizations that Sperry Rand had in mind for urbanism.

When Doxiadis invested in a UNIVAC 1107 system in 1969, its then unprecedented computing power made it suitable for commercial, scientific, and military applications. The DACC's service bureau offered classes in specialized programming languages such as Fortran (originally developed by IBM during the 1950s for scientific and engineering applications), Algol (developed in 1958 as a standard method for algorithmic description), and Cobol (designed for use in the finance and administrative systems of companies and governments), thus contributing to Greece's computer-related educational infrastructure. This was well received in Greece at a time when industry faced a shortage of trained IT personnel and few academic departments offered computer science education (Dritsa, Mitropoulos, and Spinellis 2018, 51).⁹

At a time when the diffusion of computers into the discipline of architecture was contested and many professionals were uncertain of what the computer could actually contribute to planning, Doxiadis's engagement with computer technology was underpinned by ekistics' method of systematically tackling challenges of urban planning, with the computer center both providing services beyond the architectural field and engaging in education. Thus, once the computer center was in service, it was quickly applied not only in architecture but also in the political and economic spheres, with the latter eventually justifying the computer's expense as its shortcomings for architectural work grew more apparent. The introduction of the computer was thus part of a paradox: even though the promise of more efficient planning and increased productivity was not always matched by material realization and was itself contested, the computer asserted itself as a planning tool, fostering international and interdisciplinary collaborations that enabled DA to spread its founder's ideas about ekistics and build an extensive oeuvre. Less a practical tool than a means of argument, the computer enabled Doxiadis to construct a theory or method of adaptation based on data and feedback loops. His efforts to mediate global and local needs were prompted by the computational approach, which thus aligned his method with the goals of post-war governments seeking to assert power during the Cold War. What could not be captured and modeled by the computer was largely ignored in the problem-solving designs, giving rise to the more general question of what aspects of community, the city, or the planet could be expressed through numbers and calculations.

9 In the 1960s and 1970s, computers were mainly used in the banking system. In 1954, for example, the National Bank of Greece began operating an accounting center with leased IBM equipment. These developments went hand in hand with the need to train IT staff as computer systems were adapted (Dritsa, Mitropoulos, and Spinellis 2018).

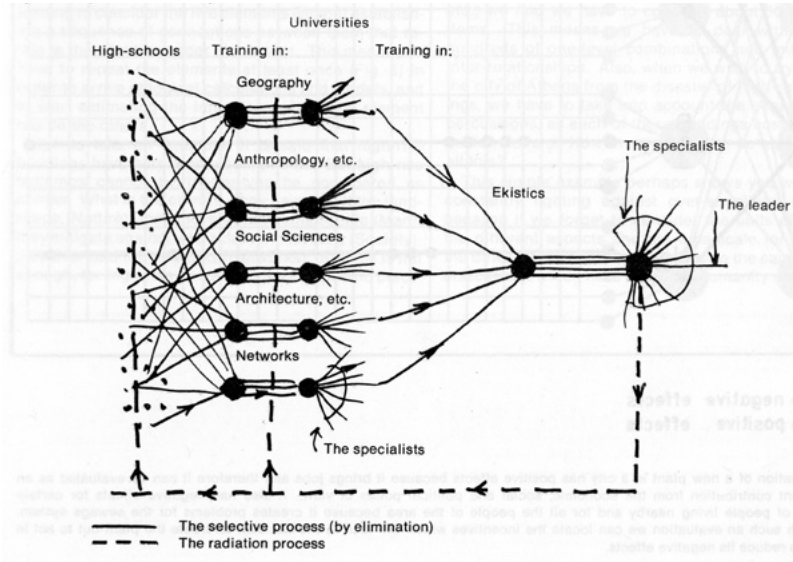


[Fig. 1] Ekistic Grid—33 million combinations with C.A. Doxiadis's signature. © Constantinos and Emma Doxiadis Foundation.

2. Tool: Grids and Diagrams

Grids and diagrams were key tools in ekistics, particularly for sorting and storing data obtained from the analysis of buildings and urban environments for future projects. Ekistics assigned the characteristics of settlements to categories and scales considered relevant for planning. The ekistics grid operated with 15 units, each having its own scale: from the individual human being to the multibillion-person Ecumenopolis—the world city. Doxiadis claimed that “all human settlements—past, present and future—can be grouped within 15 entities” (Doxiadis 1968, 393). Doxiadis further structured his method around the more general inter-relationship of five key elements: “Our subject, the whole range of human settlements, is a very complex system of five elements—nature, man, society, shells (that is, buildings), and networks. It is a system of natural, social, and man-made elements that can be seen in many ways—economic, social, political, technological, and cultural” (ibid.). Ekistics thus encompassed the inter-relationships between human beings and the environment and captured the ways in which economic, social, and political, as well as technical and cultural, factors affected them (Figure 1).

The modernist metaphor of the city as a machine whose parts interact smoothly in their functions and dynamics resonated with the systems approach. Doxiadis, however, went even further back in time by linking the



[Fig. 2] The process of training in ekistics (Education in ekistics/C. A. Doxiadis. *Ekistics*, 32, no. 193, 1971, Figure 2). © Constantinos and Emma Doxiadis Foundation.

machine metaphor to the grid plan of ancient cities as a place for storing and processing data.¹⁰ His analysis of ancient Greek cities reinforced the correlation between urban planning and management, with the grid promising easy access to how the city functioned and how it was administered. Studies of the interaction of environmental, social, and economic factors and how these affected urban areas resulted in an understanding of the city as a space dominated by infrastructures and the processing of flows.¹¹ By studying classical urbanism, Doxiadis combined the scales of the ancient Greek city with those of the machine age, touching on different scales, from the human to the industrial (Zarmakoupi 2015). Thinking about the past through data allowed him to link history with the present.

While the ekistics grid was central to the consideration of the morphological attributes of urban patterns across scales and disciplines, it was not static but subject to transformation and adaptation and thus well suited to cybernetic concepts (Doxiadis 1975; Tyrwhitt 1965). As the ekistics grid evolved, it became a cybernetic diagram where feedback loops allowed data from disciplines such as anthropology, the social sciences, and architecture to be related in the

10 The Ancient Greek Cities (AGC) research project, which ran from 1963 to 1977 and whose approach to analysis was informed by the theory of ekistics, was published in 24 volumes, and insights gained on the ancient cities were incorporated into DA's designs (Zarmakoupi 2015).

11 For the understanding of the city as being constituted by processes of information flow and processing, see Kittler (1996).

construction of ekistical knowledge. When information gathered by ekistics is interpreted by a specialist and the resultant knowledge is communicated to a leader, it can inform models of governance (Figure 2). That is, knowledge about territories is valuable when governments are trying to build an international order through resource transfers between states in the name of growth and progress (see Dutta 2013).

Margret Mead, who worked closely with Doxiadis, used the term *feedback* within the social and anthropological sciences. In the *Ekistics* journal she discussed anthropological field practice and theorized the impact of computation when stressing how inhabitants, cities, and information systems are intertwined. While Mead reflected on the position of the researcher in the process of data retrieval, the cybernetic diagrams that followed often negated the researcher's cultural background and the material facts on the ground. The specifics of a place were further negated as the data collected in the grid structure were fed into the computer, which provided statistical tables, through aggregation, offering insights into the patterns of settlements. Yet, despite the ekistics grid including the categories of social, political, and economic aspects, how projects were situated within these parameters and material contexts was not part of the equation.¹²

Despite the frictions that were part of introducing computational-based methods, the journal's authors, as well as participants in the Boston conference, embraced computational techniques that could map and visualize patterns of inhabitation and mobility. Aware that the recording of urban space would impact not only an understanding of the city but also the experience within it, ekistics further explored the visualization and interpretation of data at the intersection of computer technologies, architecture, and art.¹³

In terms of the proliferation of computing tools, activities beyond the office, bridging into other disciplines, were of interest for Sperry Rand, which had particular tasks in mind for DACC. DACC's service bureau was to promote hardware and, aside from directing sales, would conduct direct client-orientation seminars. The seminars would publicize new uses of computer technology and programs developed by DA. Education of programmers was part of the deal, too, as laid out in the program by the DACC on the future of UNIVAC in Greece (DACC 1968, 11). According to Sperry Rand, DACC's service

12 As Ali Madanipour shows in the case of Tehran, systems analysis and planning schemes that furthered a change in the social and economic conditions of the city were often at odds with the practices on the ground, as solutions, he writes, "did not grow out of the problems, as promised, but of premeditated theoretical proposals" (Madanipour 2010, 502).

13 On the role of mapping techniques in shaping and forming an understanding of politics, see Cosgrove (2008, 170–71). In 1971, the Goethe Institute in Athens hosted the exhibition *Impulse Computerart* and collaborated with ACE on an event discussing the aesthetics of the statistical side of computer mappings.

bureau was to create the hardware for clientele who would use the computer service and promote UNIVAC in the Eastern Mediterranean. From Greece, the European market was to be supplied with relatively inexpensive and flexible Greek personnel, thus serving the European UNIVAC network (*ibid.*, 12). The multifaceted approach towards computer technologies was based on a joint UNIVAC–DA programme, with UNIVAC providing the equipment and DA responsible for operations, promotion, and sales. The bureau was thus able to deal with scientific and commercial problems and also provide exhibits and training as Doxiadis built up an international network of researchers to discuss which methods and tools were best suited to ekistical topics.

In addition to collaborating with Sperry Rand, DACC engaged with foreign technical assistance programs that promoted knowledge and technology transfer through interdisciplinary planning schemes between the Global North and South. Here information became a resource, useful for governmental practices that mediated between state formation and the global market, appropriation and production, resource extraction, and manufacturing. From the grid to computer mappings, the tools for and aesthetics of visualizing statistics were part of the arguments spreading computer-based planning methods during the post-war period.

Yet, as planning paradigms promoted technologies that would map and translate human activity into planning, debate continued about how the standards or criteria of planning systems would accommodate the social organization of a community, with a central question being the extent to which urban planning would integrate the resident into the adaptive planning system and determine the self-image of a neighborhood. These frictions were also part of the urban projects supported by the Ford Foundation in Greece, which aimed to understand people by measuring and analyzing their environment, bringing together science, practice, and urban design.

3. City: Infrastructures and Communities

The concept of the “human community” was central to ekistics’ design approach and functioned as a module on which the organization of many of DA’s planning schemes was based. What constituted Greece’s informational infrastructure—the human community—was a question Doxiadis had been working on as he explored the extent, boundaries, and imaginaries of the urban neighborhood. Quantitative figures and qualitative, sociological aspects were part of the office’s studies, which focused on the size, organization, and needs of communities.

Two projects where computational technologies were used to analyze and develop planning methods based on the human community concept are the “Human Community” (HUCO) in Athens, Greece, and the “Great Lake

Metropolis" in Detroit, Michigan. These projects formed a pivot around which DA and DACC rotated in the direction of urban computing.¹⁴

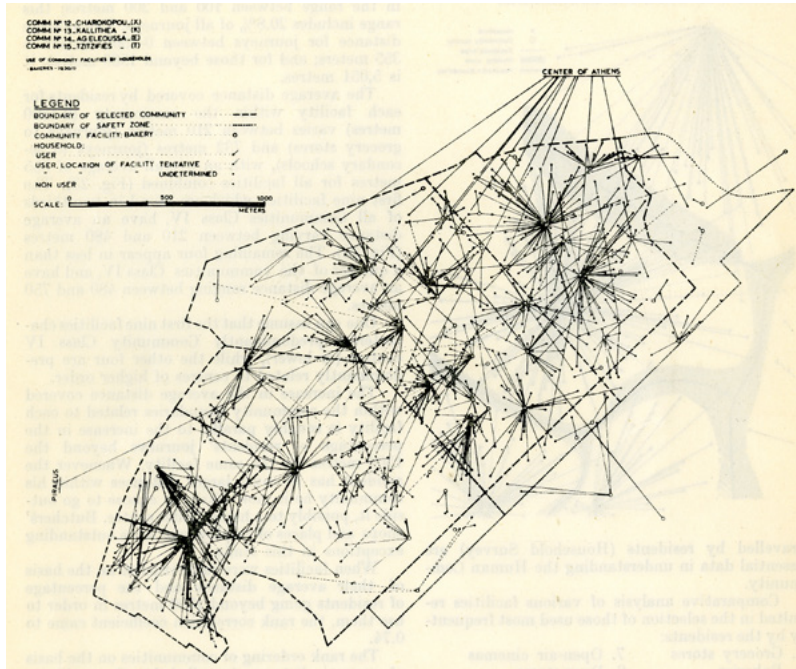
In the 1960s, in response to the urban growth of Athens, DACC became involved in the HUCO project. To identify how Athenian residents were adapting to the city's post-war changes, the project explored methods of information acquisition to monitor people's use of urban spaces. Running from 1961 to 1972, HUCO asked what constitutes a community at the level of the human scale, using the rapidly expanding city of Athens as a testing ground. The research teams carried out a systematic analysis of the city's neighborhoods, looking at urban growth and the associated economic, social, political, and administrative conditions. HUCO was co-funded by DA and the Rockefeller Foundation, which at the time supported urban design projects that bridged the gap between academia and practice.

In grappling with the urban condition of Athens, the study tested an interdisciplinary approach that drew on mathematics and the social sciences to deal with the global phenomenon of urbanization in the post-war period. Doxiadis was project director, and Demetrius Iatridis, a social scientist, was deputy project director. Applying ekistics theory to the metropolitan area of Athens, many of Doxiadis's ideas about urbanism were reiterated, and Athens communities were analyzed to gain insights into the characteristics of the smallest urban module that would allow inhabitants to meet human needs while also structuring a local neighborhood (Athens Center of Ekistics 1965, 1980).

Doxiadis believed that the essential principles of a neighborhood were stored in the locality and could therefore be distilled. As a result, HUCO's analysis of individual neighborhoods in Athens aimed to define the "DNA" of a typical urban unit. As part of the survey, interviewers trained by sociologists were sent out with questionnaires to interview residents. The results of the surveys and questionnaires were then translated into maps so that visualizations could make readable the social topography and residents' behaviors. The maps reflected the mobility of the inhabitants by documenting daily routes in relation to trade and the demand for consumer goods. These data were correlated with data on the social situation of residents, with maps documenting the extent to which residents claimed that their personal identity was linked to that of their neighbourhood (Athens Center of Ekistics 1965) (Figure 3).

In addition to the citizen interviews that were translated into graphs, data from hand-drawn maps and photo atlases, as well as computer data, were used to create a multidimensional picture of the urban fabric. Photographs accompanied the maps, and computer technology was used to collect,

14 On the role of the human community in Doxiadis's work, see Theodosis (2021).



[Fig. 3] Mapping of residents' journeys to bakeries (The Human Community. Ekistics, 20, no. 117, 1965, p. 104). © Constantinos and Emma Doxiadis Foundation.

process, and visualize the data as shown in the HUCO reports (Athens Center of Ekistics 1965).

When it came to translating the knowledge gained from the surveys into design strategies, data was folded back into the office, where the staff was modeling guidelines for future urban development. The result was endless sheets of printed numbers, which provided fodder for controversy and endless discussion.

The HUCO research team included Suzanne Keller, a sociologist from Princeton University who joined ACE from 1963 to 1965. John Virirakis, a Greek architect specializing in systems analysis, was also a member. His work exemplifies the increased importance of quantification and calculation for the city. In his paper "Minimum Effort as a Determinant of the Area, Population and Density of Residential Communities," he argues that the growth of cities can be understood as a social phenomenon arising from the people's need to achieve maximum contact, communication, and interaction with minimum effort (Virirakis 1969). In 1972, he suggested a model could be developed for the optimal distribution of community facilities throughout a city so as to minimize the energy expended by residents in traveling to use them (Virirakis

1972). While Virirakis used mathematics, charts, and graphs to relate the spatial characteristics of a neighborhood to the energy and effort expended, the conclusions he drew from the statistical surveys proved controversial.

As HUCO was attempting to capture the qualitative aspects of urban life through quantified data and behavioral patterns, criticism was voiced that the study was transforming the citizens of Athens into post-war computer data subjects (Theodosis 2015, 286). In the process, the community disappeared and then reappeared as an algorithmic principle that could be used as a blueprint for urban development. The conversion of social factors into mathematical formulae became, among other things, problematic within the HUCO project.

The question of what constitutes a planned or an unplanned community was asked by Keller, as she grappled with the neighborhood's social life (Keller 1965; Theodosis 2015, 273).¹⁵ Her report, published under the title *Neighbors, Neighboring, and Neighborhoods in Sociological Perspective* (1965), emphasized that the neighbour should be understood as a unique sociological category, while "neighborhood" encompassed a range of activities whose character and corresponding spaces had to be examined.

Keller eventually also voiced criticism of the study's approach, which viewed Athens as a pool of data divorced from its social dimensions (Keller 1971). The complex communicative and social space of the city simply could not be taken into account, and other project groups argued that the quantitative determination of interactions, paths, and distances did not translate into social interactions. Criticizing the approach, they argued that the city was more than a mix of facilities and specifications and could not be captured by computational processes (ibid., 285).

During the HUCO project, questions also arose as to how methods of data collection and processing affected the use of the physical environment and, not least, how social structures and the character of the community changed as quantification and, increasingly, economization became the basis of planning, as well as how the spatial optimization of neighborhoods and the technologies of planning intervened to change planning paradigms. The study's interest in understanding and predicting human behavior through the analysis of data also addressed how the study itself conditioned a community's sense of itself and served as a blueprint for a model city (ibid., 287).

15 The statistical techniques and mathematical models of the systems approach were generally perceived as being thorough and objective, thus making it a worthwhile decision-making tool. Controversy unfolded around the process of mapping information, which was deemed bureaucratic and failed to capture the unpredictability of human behavior and chance (see Keller 1971).

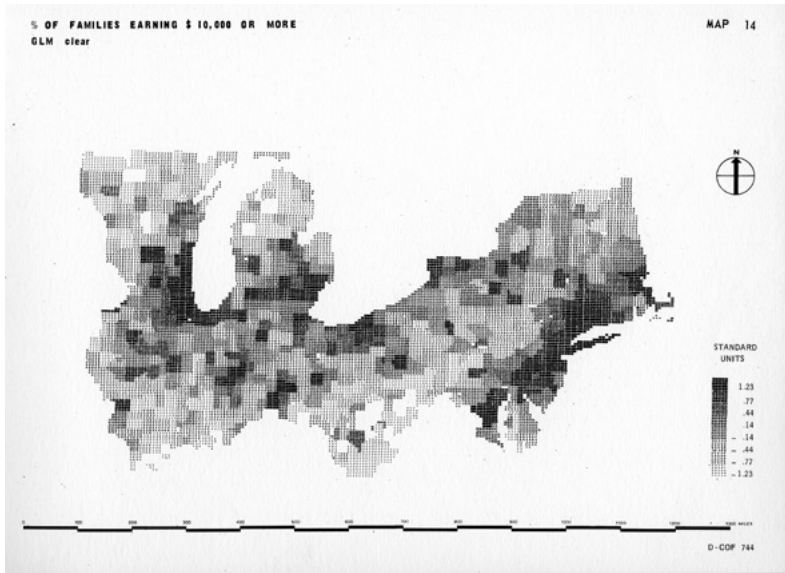
The project was deemed significant in part because the idea of an ordering urban unit could be applied to other cities. The socio-spatial effects of urbanization acting on a local context were thus part of a global concept. Consequently, the model of a community acted as a replicable medium through which not only local spaces could be modulated but also global development could be influenced.

In the mid 1960s, Detroit Edison commissioned DA to study the urban and social conditions of Detroit and to predict the growth and destiny of its environs. The office was asked to forecast the development of residential areas so that the company could plan the necessary powerline infrastructure. Beyond Detroit, the objective of the study was to model how the metropolitan area around the Great Lakes of North America would develop through the year 2000 (Leman 1976).

To forecast scenarios for a contiguous metropolitan area around the Great Lakes, DA applied several of its computer technologies. Using analytical techniques and computer programs to predict the development of the area, the office considered population shifts and urban separations caused by income, education, and race. Computer simulations took regional density into account when modeling how centers would interact with one another through infrastructure networks. Specifically, the growth of 17 states and 720 counties was studied, and the project's data were fed into a computer to be processed and visualized. The project, which ran for a five-year period, combined macro-level planning with the regional scale, and the mass of data collected resulted in the generation of 49 million alternatives (Drymiotis 2007; see Figure 4).

The shift towards a technocratic professionalism that sought to measure and represent every urban condition was questioned in the media, with skeptics asking for whom the variants of data-based planning were being generated and what interests were driving them (Schneider 1972, 332). Also, critics argued, the purported urban efficiencies and conveniences that would result from mapping urban and industrial centers, transportation networks, and travel times would actually limit citizens' access to the city. When the Cleveland *Plain Dealer* referred to the above-mentioned press release on cities planned with computers in the context of the Detroit project, the article concluded that the results of data-based planning, with mathematically calculated stuff everywhere, would be bland and dull (Colebrook 1968). Referring to DA's obsession with calculating every aspect of life and the absence of unpredictability in the optimized city, the article further implied that reducing the understanding of the city to information processing would limit the community's knowledge of the city.

However, the masses of data already collected and processed soon became an argument for further investment. Because DACC could not handle the



[Fig. 4] % of families earning \$10,000 or more GLM clear from: An application of the method of principal component analysis towards the definition of the Great Lakes Megalopolis. DOX-USA-A no. 33, Map 14 (File 25342). © Constantinos and Emma Doxiadis Foundation.

amounts of data collected for the Detroit study, DA in 1967 purchased US\$60,000's worth of computing time from the Technical University of Stuttgart (Stamatios 2022, 70).

In Stuttgart, digital maps were formatted, and information was visualized in patterns that worked with overlays and aligned urban centers according to the density of their settlements (Drymiotis 2007). When the DACC team returned to Athens from Stuttgart, DA decided to purchase a UNIVAC 1107.¹⁶ With the help of analyst-programmers, the office developed software and bespoke programming that the DACC service bureau, acting on behalf of Sperry Rand, could take advantage of.

In Greece, investment in computers combined well with the promotion of technocratic ideals that legitimized the social role of the period's planners and helped shape the nation's economic, industrial, and bureaucratic life. Because DACC acted as an independent institution, DA was able to introduce the computer to business people, journalists, and scientists and made it available

16 The cost was approximately US\$500,000 (US\$3.3 million today). At the time, approximately 10 similar computer systems were in use in the United States and only four in Europe (Dritsa, Mitropoulos, and Spinellis 2018).

for use by third parties. The National Enterprise of Electricity quickly became one of the main clients (Drymiotis 2007, 24).¹⁷

Within architecture Doxiadis's attempts to package the problems of post-war planning in the context of computation gave rise to the graphic language of bars, charts, maps, and graphs that soon saturated the DA office and coordinated its communications. Although the use of the computer in architecture was controversial—the limits of predictability were frequently highlighted, and projects often abandoned—the corporate and political entities that were part of the planning schemes ensured the continuation of computational approaches. During the post-war period, DACC amplified the planning methods of the Cold War era, thus leading to urban reconfigurations and geopolitical realignments and ensuring the persistence of its methods into the present day.

4. Planet: Resources and Habitats

"On the Linkage of Socio-economic and Ecologic Systems"—published in the *Ekistics* journal in 1969—deals with the computer as a tool capable of meeting the needs of planning in a more comprehensive and ecological way. The authors, who were associated with the Regional Science Research Institute and Harvard University, focus on the interdependencies of food chains and recreational activities and how they affect the economy within the Plymouth Bay region of Massachusetts (Isard et al. 1969).¹⁸ Although computational methods were not applied in practice due to the immaturity of the computer technology, ideas from systems theory and cybernetics informed the studies that focused on the interdependencies between the parameters that affected ecological and economic systems.

Using an input-output scheme that served as a template for cybernetic methods of planning, the study focuses on understanding the interdependencies among the parameters that make up the region's food chains. A large salt marsh forms an important resource, producing detritus and plants, which in turn are the food base for annelids, algae, molluscs, crustaceans, and so on. The latter in turn ensure the food base for winter flounder, important for the

- 17 Drymiotis writes, "Periklis Valvis would come almost every day with his Fortran programs (boxes full of perforated cards), run to the 1107, pick up the results and go back to his office to elaborate them." (Drymiotis 2007, 24). In 1984, DACC was separated from DA and became a new company, Delta Informatics, which in 2000 represented Sperry Univac in Greece under the name Unisystems; it was eventually bought by First Data (Dritsa, Mitropoulos, and Spinellis 2018).
- 18 The American economist Walter Isard, who promoted and developed the Regional Science Association, was interested in the systems of cities and regions, looking beyond the city to economic regions, countries, and the planet (see Boyce 2010). Regional science entered British planning through Brian McLoughlin's "systems approach" (McLoughlin 1969).

local economy. Situated at the end of this sequence of organisms, where each element derives energy from the consumption of the previous one, the winter flounder define the output of the ecosystem. In addition, as an input to both commercial and recreational fisheries, winter flounder is the link between the ecosystem and the region's social and economic systems (Isard et al., 1969). The study thus establishes a link between an ecosystem and a social system, where the former provides an import (input) to the latter. The input-output format is further used to illustrate another type of linkage—namely, the way in which the social system provides an import to the ecological system. Water pollution, an output of society, becomes an input to the ecological system, which in turn functions as an input by providing natural resources that are used by people for recreational purposes. The feedback system allows for a better understanding of the interdependencies among selected natural resources and recreational activities and their importance for development strategies (*ibid.*).

The article argued that as many activities and resources as possible must be considered when evaluating ecological and economic systems. Furthermore, planning had to strive for a more complete material balance when considering economic, social, natural, and other activities. The aim of the authors was also to identify more precisely the existing interdependencies of a multi-regional socio-ecological system so as to provide expertise for policy-making at a political level. Information processing and systems thinking were thus well suited to fostering design methods that would incorporate and relate the various factors.¹⁹

As Doxiadis extended his expertise in planning through research and international collaborations and engaged in transnational practice, the global ecological perspective, as Panayiota Pyla argues, increasingly became part of Doxiadis's vision of environmentalism (Pyla 2002). Pyla shows how the concept of ekistics, which during the post-war period inter-related people and the environment to develop schemes for reconstruction, would increasingly turn towards ideas of environmentalism, and how, in line with these developments, the ekistics grid evolved (Pyla 2009, 7). This underlines once again how the development of tools, the ekistics grid, anticipated the systems thinking and allowed Doxiadis to operate at different scales as he sought to relate the local to the global.

19 Regional science was founded in the late 1940s after campaigns organized by Isard finally led to the founding of the Regional Science Association in 1954. In the same vein, Doxiadis promoted ekistics by launching his journal, founding ACE, and organizing the Delos Symposia. Nevertheless, as Pyla notes, "for all its emphasis on embracing different scientific disciplines, ekistics' engagement with other disciplines was rather vague" (Pyla 2008, 44–45).

When the United Nations held its first Habitat Conference in 1976, it stressed the need to understand human settlements not only in terms of cities and towns but also in the context of global issues such as the environment, population, food, and the world economic order. The conference report highlights inequalities among countries—including disparities in access to life-supporting resources such as air, water, and land—and proposes that solutions would be found in the appropriate use of science and technology and that governments and the international community would facilitate the necessary technology transfers. The report also addresses tensions between global planning and local conditions, noting that technology transfer and infrastructure should be compatible with local resources, culture, and conditions. Finally, it emphasizes that the development of technologies to reduce waste should incorporate existing technological systems, taking into account the design and selection of materials that would become waste (UN-Habitat 1976).

Doxiadis, when grappling with the growth of cities and settlements and a global scale, in addition to engaging in computational strategies, coined several terms to capture his view of planning on a planetary scale, including *Ecumenopolis*, a city that would eventually cover the entire planet (a worldwide urbanization pattern); *Ecumenokepos*, a global garden that would work in concert with the global city; and *Entopia*, the idea of being “in place.” With the idea of ecological world settlement and the concept of *Ecumenopolis*, Doxiadis predicted that the world’s existing megalopolises would expand to form a single, globe-covering and globally interconnected urban network (Doxiadis 1961, 1968). As Pyla shows, with the concept of *Ecumenopolis*, ekistics, in addition to fostering international development, prompted architects to manage and protect the environment on a global scale (Pyla 2009, 10).

The world city was the epitome of Doxiadis’s modernist ambition to coordinate the entire system of knowledge about the physical environment to respond to the totality of human needs across cultural, geographic, and socioeconomic differences. Yet, his belief that environmental protection could be managed through the global perspective of *Ecumenopolis* was also criticized for being too general and for overestimating the ability of architects to deal with the uncertainties of the future. Alternative visions of *Ecumenopolis* were formulated, too, including Fuller’s suggestion that increasing mobility would be a more powerful force in holding the world together than the static settlements envisioned by Doxiadis.²⁰ This was in line with McLuhan’s vision of the electronic age, where spatial and temporal barriers would be overcome by a new kind of interconnectedness through electronic networks. Far from the human scale defined by Doxiadis’s designs, McLuhan’s ideas of networks were set in opposition to it (McLuhan 1963, 257).

20 On the different views on *Ecumenopolis* that were discussed at the Delos Symposium, see Pyla (2009, 13).

Doxiadis's optimism for a post-political, global society in which social, economic, racial, and ethnic inequalities could be managed away by benevolent technocrats negated the political conflicts and power influences that drove the planning policies. From its beginnings as a development consultancy, DA had appealed to its international clients (often the governments of young nations) by promising socio-economic modernization. The firm's global perspective and computer-based approach allowed it to carry out projects around the world. This global view, conditioned by the growth of cities in the Global North and Global South, promoted urban industrialization and the management of land and resources, with architects acting as managers in the field.²¹

The adaptation of the computational thinking thus oscillated between immaterial networks and material conditions found on the ground. These planning concepts were situated within the interdependencies connecting political interests and industrial growth and in turn were linked to concepts of environmentalism. The role of the architect as an expert who could synchronize local and global solutions was characterized by a technocratic logic that overrode political debates at a time of nation building driven by post-colonial agendas.²²

5. Methodological Appeal of the Computer

In the 1960s Doxiadis was known as "the world's busiest remodeler." He was featured in *Life* magazine standing on what was called a "study machine," a mobile cage that carried observers over large-scale models of the projects the office was working on. The article emphasizes that the development of tools was central to understanding design problems: in this case, the machine would allow cities to be studied "from every possible angle, not as a drawing or an image, but as a complex system in four dimensions" (Anonymous 1966, 56). The article further examines the scales and methodologies of post-war urban planning and reiterates Doxiadis's role in shaping social, political, and economic realities worldwide. And even though the image shows the architects standing over an analog model, the reference to modeling and controlling various aspects of society, to take on a global perspective, alludes to the statistical techniques and mathematical models of the ekistics systems approach.

21 On the export of Greek bauxite after the country's integration into the Marshall Plan and in response to the technocratic elites' promotion of the exploitation of natural resources, especially minerals, to build a strong industrial sector, see Papastefanaki (2013).

22 On the role of architects and planners in the formation and transformation of new nation-states after the Second World War, and the associated shifts in cultural and economic imaginaries, see the contributions to *Aggregate Architectural History Collaborative* (2022).

The post-war research in computer technology, based on cybernetic concepts, fit well into this picture, with data-based planning giving power and legitimacy to architects who increasingly acted as policy advisers. The abstract world of modeling, made up of symbolic machines, thus had a lasting effect on the discipline of design, which had previously been dominated by intuitive designs and the idea of pure material reification. Planning in the context of computation was on the other hand increasingly seen as a means of regulating and controlling the challenges of urban growth while shaping processes of globalization.

Architecture, in turn, proved to be one of the fields in which computer companies engaged. Sperry Rand, in particular, assigned to DA the role of introducing computing into various areas of Greek society and the Mediterranean. Although the role the computer would play in architecture was not clear from the outset and was thus subject to debate, the field embraced the technology and explored planning algorithms. The computer fit well with contemporaneous efforts to understand urban settlements and city planning through data.

In the post-war period, DA's methods were generally perceived to be thorough and objective, allowing it to work in a managerial capacity when mapping and planning cities. The holistic view offered a way of dealing with the problems of growing cities, and the rational techniques became entangled with the politics that operated through architecture, promoting and controlling modernization. Even if the systems analysis that promoted change in social and economic conditions was often at odds with practice on the ground, as the articles in the journal *Ekistics* show, the overarching theoretical approach ensured that computing became an established tool in the field of architecture. This was associated with the social, ecological, and economic adaptation of systematic approaches.

Controversies over what the computer would bring to architecture were part of the embrace of the computer by architecture, and contributions to *Ekistics* drew on the methods and tools deployed in other fields where computers had been applied. The tools and methods discussed in the *Ekistics* issue, as well as the role technology played in facing the challenges of the post-war period, are the ancestors of today's data-based planning. The ekistics grid and the data-based methods were indicative of the efforts to control places. When trying to address urban growth and its dynamics, ekistics, like systems theory, developed tools and methods that could easily be applied to other fields and other places. As such, the science of ekistics was part of a more general shift within urbanism from modernist to systemic approaches. Ekistics emphasized networks and global interconnectedness and used technology to record and evaluate its theories (Wigley 2001). Moreover, ideas of change and growth without boundaries and ideas of technology capable of solving every possible

problem, from demographic growth to energy shortage, from pollution to economic backwardness to ethnic and social unrest—all these made Doxiadis's vision the perfect vehicle for the ideology of governments and humanitarian organizations both.

The issues of the journal and Doxiadis's own work show that, as the computer was integrated into architecture, it passed through distinct methodological stages: from a tool for processing and analyzing large amounts of data to a modeling tool that could render and animate urban dynamics. The computer was conceived as a medium for collaboration and the exchange and distribution of information. While many of its actions were carried out in an analog manner, some began to imagine that the computer could one day act as a thinking machine, capable of artificial intelligence by working with pattern recognition.

The methodologies that fed into the embrace of the computer by ekistics, with its focus on the management of human settlements using data-based and information systems, might offer frameworks for understanding the situatedness of computational practices within contemporary planning. To reflect on the role of the computer within ekistics—and the politics and optimism associated with it—is also to reflect on how discourses of modernization were carried forward through computational techniques during the post-war period. As the notion of design shifted from the discrete object towards a set of practices, the collection, processing, and visualization of data took on a central role, with the aim of optimizing urban life through planning. The understanding of planning was thus increasingly entangled with the notion of problem-solving and the development of tools for managing the city and the planet.

This text arose from a research stay at the Institute for Advanced Study on Media Cultures of Computer Simulation (MECS) at Leuphana University Lüneburg. I would like to thank my fellow colleagues and the editors of this issue for the ongoing discussions and Giota Pavlidou, the archivist of the Doxiadis Archives, for her assistance during my research stay at the archives.

References

- Aggregate Architectural History Collaborative, ed. 2022. *Architecture in Development: Systems and the Emergence of the Global South*. Abingdon and New York: Routledge.
- Anonymous. 1966. "City Planner Constantinos Doxiadis: Busy Remodeler of the World." *Life*, October 7, 1966: 55–60.
- Athens Center of Ekistics. 1965. "The Human Community." *Ekistics* 20, no. 117: 83–113.
- . 1980. "HUCO: The Human Community in Athens." *Ekistics* 47, no. 283: 232–63.
- Bailey, Grayson, Nathalie Bredella, and Carl Steinitz. 2025. "Is That a Computer Program? Carl Steinitz on the Early Years of Harvard's Laboratory for Computer Graphics and Spatial Analysis." *Dimensions Journal of Architectural Knowledge* 4 (7): 77–98.
- Biographical note. n.d. Constantinos and Emma Doxiadis Foundation. Accessed November 2024. <https://www.doxiadis.org/ViewStaticPage.aspx?ValueId=4276>.

- Boston Architectural Center. 1964. *Architecture and the Computer. Proceedings, First Boston Architectural Center Conference*. Boston, MA: Boston Architectural Center.
- Boyce, David. 2010. "Walter Isard, 1919–2010." Regional Science Association International. Accessed November 2024. https://www.regionalscience.org/images/PDF/Walter_Isard_obituary.pdf.
- Colebrook, Paul. 1968. "Greek Planners Calculate a Computable Future." *Plain Dealer* (Cleveland), December 8.
- Cosgrove, Denis. 2008. *Geography and Vision: Seeing, Imagining and Representing the World*. London: I.B. Tauris.
- DACC. 1968. "The Future of UNIVAC Computers in Greece and the program suggested by Doxiadis Associated Computer Center (DACC)." Ref. Code C-DCC 2506, Doxiadis Archives.
- De Dominicis, Filippo. 2019. "To Survey, Control, and Design Doxiadis and Fathy on Africa's Future and Identity: 1959–1963." In *Routledge Handbook of Urban Planning in Africa*, edited by Carlos Nunes Silva, 193–208. London: Routledge.
- Derry, John R., Jerome W. Lubin, Onalee M. Laird, Clyde Carrell et al. 1969. "An Information System for Health Facilities Planning." *Ekistics* 28, no. 164: 19–24.
- Doxiadis, Constantinos. 1961. *Ecumenopolis, the Settlement of the Future*. Research Report. Athens: Athens Technological Organization.
- . 1966. *Between Dystopia and Utopia*. London: Faber and Faber.
- . 1968. *Ekistics: An Introduction to the Science of Human Settlements*. London: Hutchinson.
- . 1972. *Architectural Space in Ancient Greece*, translated and edited by Jaqueline Tyrwhitt. Cambridge, MA: MIT Press.
- . 1974. "Ekistics and Mathematics." *Ekistics* 37, no. 222: 305–308.
- . 1975. "Economics and the Ekistic Grid." *Ekistics* 40, no. 236: 1–4.
- Doxiadis, Constantinos, and Club of Rome. 1970. "Model (The Club of Rome)." In C.A. Doxiadis Archives, Archive Files, C.A. Doxiadis Personal, 07. General Scientific Material, Gen Series, Doxiadis' Lectures/Articles/Papers (Gen 1001). Accessed November 2024. <https://www.doxiadis.org/ViewDocument.aspx?Id=2325359>.
- Dritsa, Konstantina, Dimitris Mitropoulos, and Diomidis Spinellis. 2018. "Aspects of the History of Computing in Modern Greece." *IEEE Annals of the History of Computing* 40, no. 1: 47–60. <https://doi.org/10.1109/MAHC.2018.012171267>.
- Drymiotis, Alex. 2007. "C. A. Doxiadis and Information Technology: His Contribution to its Development in Greece." *Ekistics* 74, no. 442–447: 22–26.
- Dutta, Arindam, ed. 2013. *A Second Modernism: MIT, Architecture and the "Techno-Social" Moment*. Cambridge, MA: MIT Press.
- Eberstadt, Nicholas. 1997. "The United Nations' 'Development Activities': What Impact on Third World Development?" *World Affairs* 159, no. 4: 151–67.
- [Editorial.] 1969. "Computers in the Service of Ekistics." *Ekistics* 28, no. 164: 1–3.
- Fthenakis, E. 1964. "Computer Mapping." Document Reference Code 22342, Doxiadis Archives.
- Fuller, Buckminster. 1965. "Why I Am Interested in Ekistics." *Ekistics* 22, no. 131: 180–81.
- . 1969. *Utopia or Oblivion: The Prospects for Humanity*. Toronto: Bantam Books.
- Isard, Walter, Kenneth Bassett, Charles Choguill, John Furtado, et al. 1969. "On the Linkage of Socio-economic and Ecologic Systems." *Ekistics* 28, no. 164: 28–34.
- Jansen, Wim. 2009. "Neurath, Arntz and ISOTYPE: The Legacy in Art Design and Statistics." *Journal of Design History* 22, no. 3: 227–42.
- Keller, Suzanne. 1965. "On Neighbours, Neighbouring and Neighbourhoods: Fact and Theory." Document, Ref. Code 19427, RR-ACE:68 (105), 2.
- . 1971. "The Non-Local 'Community' in HUCO." Document, Ref. Code 22702. Doxiadis Archives.
- Kittler, Friedrich. 1996. "The City Is a Medium," translated by Matthew Griffin. *New Literary History* 27, no. 4: 717–29.
- Kotsioris, Evangelos. 2015. "Architecture and the Computer: A Contested History." *Architectural Review*. Accessed November 2024. <https://www.architectural-review.com/archive/architecture-and-the-computer-a-contested-history>.

- Kotsioris, Evangelos. 2022. "The Computer Misfits: The Rise and Fall of the Pioneering Laboratory for Computer Graphics." In *Radical Pedagogies*, edited by Beatrice Colomina, et al., 314–17. Cambridge, MA: MIT Press.
- Kyrtsis, Alexandros-Andreas, ed. 2006. *Constantinos A. Doxiadis: Texts, Design Drawings Set-lements*. Athens: Ikaros.
- Leman, Alexander B. 1976. "Great Lakes Megalopolis—Canada," *Ekistics* 41, no. 243: 114–19.
- Madanipour, Ali. 2010. "The Limits of Scientific Planning: Doxiadis and the Tehran Action Plan." *Planning Perspectives* 25, no. 4: 485–504.
- Maynard, Michael. 1968. "Cities of Tomorrow Planned on High-Speed Computers," UNIVAC News, Reference Code 22825, Doxiadis Archives.
- McHale, John. 1962. *R. Buckminster Fuller*. New York: Georg Braziller.
- McLoughlin, John Brian. 1969. *Urban and Regional Planning: A Systems Approach*. London: Faber and Faber.
- McLuhan, Marshall. 1963. "Contributory Papers." *Ekistics* 16, no. 95: 257.
- McMahon, Catherine. 2013. "Predictive Machines: Data, Computer Maps, and Simulation." In *A Second Modernism: MIT, Architecture, and the "Techno-Social" Moment*, edited by Arindam Dutta, Stephanie Tuerk, Michael Kubo, Jennifer Yeeseu Chuong, and Irina Chernyakova, 436–73. Cambridge, MA: MIT Press.
- Mead, Margaret. 1969. "Research with Human Beings: A Model Derived from Anthropological Field Practice." *Ekistics* 28, no. 164: 9–13.
- Meadows, Donella. 1972. *The Limits to Growth: A Report for the Club of Rome's Project on the Predicament of Mankind*. New York: Universe Books.
- Minski, Marvin. 1964. "Panel Discussion." In *Architecture and the Computer: Proceedings, First Boston Architectural Center Conference*, 42–51. Boston, MA: Boston Architectural Center.
- Papastefanaki, Leda. 2013. "Greece Has Been Endowed by Nature with This Precious Material: The Economic History of Bauxite in the European Periphery, 1920s–1970s." In *Aluminum Ore: The Political Economy of the Global Bauxite Industry*, edited by Robin S. Gendron, Mats Ingulstad, and Espen Starli, 158–84. Vancouver: University of British Columbia Press.
- Pyla, Panayiota. 2002. "Ekistics, Architecture and Environmental Politics, 1945–1976: A Prehistory of Sustainable Development." PhD dissertation, Massachusetts Institute of Technology, Cambridge, MA.
- . 2008. "Baghdad's Urban Restructuring, 1958: Aesthetics and Politics of Nation-Building." In *Modernism and the Middle East: Architecture and Politics in the Twentieth Century*, edited by Sandy Isenstadt and Kishwar Rizvi, 97–115. Seattle: University of Washington Press.
- . 2009. "Planetary Home and Garden: Ekistics and Environmental-Developmental Politics." *Grey Room* 36: 6–35.
- Schneider, Jerry B. 1972. "Plan Review: Doxiadis' Detroit: Forty-Nine Million Alternatives." *AIP Journal* 38, no. 5: 332–33.
- Scott, Felicity. 2016. *Outlaw Territories: Environments of Insecurity/Architectures of Counter-insurgency*. New York: Zone Books.
- Shoshkes, Ellen, and Sy Adler. 2009. "Planning for Healthy People/Healthy Places: Lessons from Mid-twentieth Century Global Discourse." *Planning Perspectives* 24, no. 2: 197–217.
- Stamatios, Schizakis. 2022. "Plugged In: The Introduction of New Media Art Practices in Greece in the 1970s and 1980s." PhD thesis, University of Sunderland, UK. Accessed November 2024. <http://sure.sunderland.ac.uk/id/eprint/14699>.
- Steinitz, Carl, and Peter Rogers. 1969. "A Systems Analysis Model of Urbanization and Change: An Experiment in Interdisciplinary Education." *Ekistics* 28, no. 164: 60–70.
- Taylor, John, and K.R. Carter. 1969. "Some Instructional Dimensions of Urban Gaming-Simulations." In "Computers in the Service of Ekistics." *Ekistics* 28, no. 164: 54–59.
- Theodosios, Lefteris. 2015. "Victory over Chaos? Constantinos A. Doxiadis and Ekistics 1945–1975." PhD thesis, Universitat Politècnica de Catalunya, Barcelona.
- . 2021. "Systemic Methods and Large Scale Models in Ekistics." *Nexus Network Journal* 23: 171–86.

- Tsiambaos, Kostas. 2012. "Isotype Diagrams from Neurath to Doxiadis." *Architectural Research Quarterly* 16, no. 1: 49–57.
- Tyrwhitt, Jaqueline. 1965. "The Ekistics Grid." *Architectural Association Journal* 87: 12.
- UN-Habitat. 1976. *Report of Habitat: United Nations Conference on Human Settlements, Vancouver, Canada, 31 May to 11 June 1976*. A/CONF.70/15. New York: United Nations. Accessed November 2024. <https://digitallibrary.un.org/record/793768?ln=en&v=pdf>.
- Virirakis, John. 1969. "Minimum Effort as a Determinant of the Area, Population and Density of Residential Communities." *Ekistics* 27, no. 163: 362–71.
- . 1972. "Minimization of Energy as Determinant of the Grouping of Community Facilities." *Ekistics* 33, no. 199: 503–11.
- Wigley, Mark. 2001. "Network Fever." *Grey Room* 4: 82–122. <https://doi.org/10.1162/152638101750420825>.
- Zarmakoupi, Mantha. 2015. "Balancing Acts between Ancient and Modern Cities: The Ancient Greek Cities Project of C.A. Doxiadis." *Architectural Histories* 3, no. 1: 1–22.

HISTORY OF COMPUTING

LOGIC

CONCURRENCY

HISTORY OF SCIENCE

PHILOSOPHY OF SCIENCE

Tense and Temporality: Computing and the Logic of Time

Troy Kaighin Astarte

This chapter explores the role of time in logic, from ancient history to modern computing. It provides an outline and primer on the development of tense logic, and explores how these ideas found their way into computer science. The chapter examines how certain philosophical problems in this space saw new light as technical questions, and explores the role of logic in computer science.

1. Introduction

The consideration of time in logic is both ancient and thoroughly modern. From Greek philosophy to computer science, investigation into tensed logical statements showed they increase the expressiveness of a logic and its connection with the real world, but also its complexity. While the classic Aristotelian view was that a statement could change its truth value with time, this became theologically concerning when examining statements about determinism and the omniscience of God. Debate on the logic of time saw detailed and thorough treatment only in the 1950s when A. N. Prior developed “tense logic,” a form of modal logic¹ in which notions of time and contingency

1 This branch of logic concerns predicates which express possibility or certainty, typically through “possibly” and “necessarily” operators. Such logics are often evaluated by

are expressed as modalities. Although Prior saw an immediate application to computing, the ideas broke into this field only in the 1970s, and via alternate means.

In the late 1960s Zohar Manna was working on systems for proving termination of programs. Rod Burstall took some of Manna's ideas and gave them a different notation in a 1974 paper—and made the connection to modal and temporal operators. Meanwhile, Amir Pnueli, who had worked alongside Manna and Nissim Francez on cyclical programs, took Burstall's ideas and applied them to concurrent behavior, naming this “The temporal logic of programs.” Concurrent systems are those in which different components act simultaneously while sharing certain resources; they are mandated by the presence of hardware which operates at different speeds, and concurrent behavior may be deliberately introduced into programs for performance benefits or to model real-world parallel systems. Handling the behavior of such programs has been recognized as a difficult task since at least the 1950s,² and the 1970s saw the emergence of various theoretical frameworks—of which temporal logic is one—for modeling and reasoning about concurrent behavior.

The case of temporal logic demonstrates an example of ancient ideas considered purely philosophical ending up with serious practical applications. One interesting facet of this is the rediscovery of many old problems in new contexts. Computer scientists working with temporal logic came up against and argued about issues such as whether time should be considered branching or linear; the interpretation of future tensed propositions in the present; and the creation of appropriate models for logical systems. In most cases, these computer scientists considered these problems while being entirely unaware of the history behind them.

This chapter explores the “logic” background of temporal logic and investigates how the ideas came into computing. It considers the early field of temporal logic and shows examples of old arguments recurring in new contexts. It begins with a brief overview of some pre-modern views of logic in time, before considering some areas of particular interest that re-emerge in the computing era. Prior's tense logic is introduced, and then the routes by which these ideas made their way into computer science literature is examined. Temporal logic for concurrency is then discussed and the earlier

means of a “possible world” semantics which allows the linking of modal propositions. Neither the logic itself nor its history is discussed in detail here; the interested reader is referred to Hughes and Cresswell (1968) as a standard reference or Goldblatt (2006) for its history.

- 2 The major challenge is preventing harmful interference, which can arise when different concurrent agents access a resource simultaneously. See Astarte (2023) for discussion of the historical emergence of this problem and some of the ways to address it.

problems are reintroduced; the chapter concludes by considering what this story says about the role of logic in computer science. The case is made that temporal logic, like much of theoretical computer science, sits neither in logic nor mathematics, nor within the practical toolset of the everyday programmer, but somewhere else entirely.

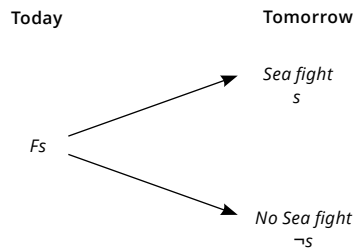
A large part of the material in this chapter on the pre-computing era draws from the analysis by Øhrstrøm and Hasle (2007), who provide both contextually accurate and modern rephrasing of historical logic systems. A final note on terminology: in this chapter, “tense logic” is taken to refer to the logic of Prior and others, working outside a computing context; “temporal logic” will denote such logic ideas applied to computing.

2. Old Problems, Old Logics

An early problem, discussed in the work of Aristotle, was how to cope with the inherent unknowability of the future (Øhrstrøm and Hasle 2007, §1.1). In “On Interpretation” (Chapter IX), the example is presented of a potential fight happening at sea tomorrow. We use *Fs* to denote the statement “There will be a sea fight tomorrow,” for reasons which will become apparent later. The question is: How can the truth of *Fs* be interpreted today?

If there are two potential outcomes which are opposite—a fight, or no fight—and neither of these is contingent on anything today, but everything which happens (i.e., every true statement) is necessitated, then there is no way to determine right now whether *Fs* or its negation is true. Aristotle recognized the modality of this situation: If tomorrow there is indeed a fight, making *Fs* true yesterday, did that make it also necessary? And if tomorrow there is no fight, was *Fs* not possible yesterday? Aristotle’s perspective was that the past and present are deterministic, but the future is not. All true statements about the past and present are necessary, but true statements about the future might only be possible. This, however, created a disquieting asymmetry between past and future.

One way to interpret Aristotle’s position was reasoned out by Richard of Lavenham (c. 1380) starting with the idea that it might simply be impossible to interpret statements like *Fs* (Øhrstrøm 1983). To many religious logicians, like Lavenham, this was problematic due to God’s omniscience and the future being deterministic was equally unappealing. Lavenham’s solution was to reject the necessity of the past. An alternative approach, favored by Jan Łukasiewicz (1873–1956), was to reject the law of the excluded middle, and introduce a third truth value, “undetermined,” which could be the result of evaluating statements like *Fs* (Øhrstrøm and Hasle 2020).



[Fig. 1] Aristotle's sea fight example

More relevant for the current chapter, another interpretation of Aristotle was put forward by William of Ockham, and later Peirce, which allowed both the excluded middle for future propositions, and non-determinism of the future. These were reasoned out into full modern logical systems by Prior (1967). The idea was to reject the classical view of time as a single line.

The alternative was to view the future as branching out from the present like a tree, in a series of different possibilities. A simple illustration of the sea-fight branch can be seen in Figure 1. The present can even be considered as one of a set of potential presents along various parallel lines of time, no longer accessible due to previous forks. As time progresses, one of the possible futures becomes the present, and then the past, leaving an asymmetric tree with the past as a line.

Using this branched notion of time allowed Prior to develop a few ways to address the knowability of future events. One is that all future paths are true until they reach the present and are evaluated, though Prior didn't care for that one. After Ockham, the most religiously satisfying view is that only one future is true but is as yet unknowable to mere humans. Finally, after Peirce, another interpretation of future propositions is that they are true only when true in all possible futures. Branching approaches like these, especially the second, allowed logicians to solve the omniscience problem: while to a human the various potential paths might be unknown and therefore subject to free will, God is able to identify which future is the real one.

Prior's contributions to the logic of time are not confined to the branching structure thereof; the next section examines his work and sources of inspiration.

3. Prior and Before

Arthur Norman Prior (1916–1969) was born in New Zealand and brought up a devout Christian, even intending to follow a career as a Presbyterian minister while he studied philosophy at the University of Otago (Copeland 2020). Prior's early interests lay in ethical philosophy, and the relationship between free will and omniscience. He was introduced to logic by one of his tutors, John Findlay. Prior favored realism over idealism and was opposed to purely formal logic in philosophy, writing that logic was useless unless it helps describe the world (Jakobsen 2020). After a little work on ethical logic, Prior became interested in the history of logic. Likely inspired by Findlay, who had sketched some propositions for a calculus of time, Prior began to study, among others, Peirce, Łukasiewicz, and Boole.

Boole had written about time in the context of probability, assigning “simple” propositions a numerical value between 0 and 1, combining them with algebraic laws to create more complex events (Øhrstrøm and Hasle 2007, §2.1). However, he introduced an interesting “surreptitious” shift in which these values represented not the probability of the events occurring, but rather the truth of statements that those events occurred (Durand-Richard 2023). This allowed him to sidestep the philosophically challenging problem of ascertaining the true independence of events. Prior shared Boole's belief in the importance of finding a role for time in symbolic algebra, though he felt Boole's logic, lacking specific operators for tensed statements, was insufficiently rich (Prior 1957, Appendix A).

Like the medieval scholars he had studied, Prior believed that since human speech naturally contains many tensed statements, any system of logic which purports to describe the world as experienced by humans must be equally rich in its ability to express temporal notions. Critically, Prior held that sentences like “Socrates is sitting” must be viewed as complete without needing extra components such as “at time *t*” to become interpretable (Øhrstrøm and Hasle 2007, §2.5).

Prior's approach here came from his historical knowledge, especially Peirce and Łukasiewicz. Peirce too had studied medieval logic and used concepts from this in his semiotics, though he believed that the early 20th century was not the right time to bring temporal aspects back into logic (ibid., §2.2). Peirce related modality with time by connecting “actuality” with the past and present, and “possibility” and “necessity” with the future, which enabled his view of the future as a set of branching possibilities, and his proposal for rationalizing the apparently competing doctrines of human consciousness, omniscience, and determinism.

Łukasiewicz's work in the 1920s and 30s had established a new "Polish" school of logic, cementing the logical paradigm in which Prior later worked. One relevant facet of that approach was the translation of historical logical systems into symbolic ones, thereby allowing their properties to be analyzed alongside modern logic. Just as Prior would, Łukasiewicz used concepts from ancient and medieval logic in his original work (Simons 2021). This was present in his interpretation of Aristotle as describing a contingent future view that rejected bivalence in favor of a third, "possible," truth value. Though Prior did not himself favor trivalence, he was full of praise for Łukasiewicz in many other ways, and made extensive use of his Polish notation.³

Armed with this inspiration, Prior developed a number of systems he called "tense logic" based on the ordering of events in time. He took the universe of possible worlds from modal logic into the temporal domain as instants in time and the accessibility relation between these instants became sequentiality. In this context, the modal "necessity" operator became a temporal "always," and "possibility" became "at some point." Prior presented his work in the John Locke series of lectures in Oxford over 1955–56, and published a book the next year (Prior 1957). By 1967, the ideas had caused significant waves in the logic community (Copeland 2020) and Prior had written a sequel, in which he presented a variety of logical systems and their subsequent properties (Prior 1967).

In the present chapter we will consider the basic aspects of Prior's tense logic and some specific points, rather than attempting to cover every single system.⁴ Tensed statements are expressed as logical propositions combining untensed terms, usually lowercase Roman letters, with uppercase Roman letter operators capturing the tense. P and F indicate that something happened at some point in the past or future (respectively) and are outlined in Figure 2a. H and G capture events that have happened, or are going to happen, continuously; they are presented in Figure 2b. These statements are interpreted with reference to a privileged instant which is "now."

This system also does not allow the expression of duration. In order to study this, we first need to examine another critical problem in the logic of time: the basic units of the logical system.

Though Prior (1967) discussed this idea in some depth, examining the consequences of various choices, it has a longer pedigree, much like the question of linear or branching time. As far back as Aristotle, again, the question

3 This concise system for writing propositions uses only the Roman alphabet but is deeply expressive and has the benefit of unambiguous order of interpretation without requiring brackets. For more on the way this writing tool shaped logic, see Dunning (2020).

4 The interested reader is referred to Øhrstrøm and Hasle (2007) for extensive discussions and many fine points.

The operator P indicates the past and F future (relative to fixed *now*); let r = 'it rains' (either in the present, or untensed).

'It will rain': Fr

The following graph presents a visualisation of some values that satisfy the above equation.



We don't know when it will rain, only that at some point it will; we don't know when or even if the rain will stop.

Operators can be combined to create more complicated tensed statements such as 'It looked like it was going to rain': PFr



As seen in the graph, the rain could already have started by now; equally the start of the rain could still be in the future.

Such tensed logic statements can be combined with standard propositional operators; introducing w for wet grass, 'The grass is wet, so it has rained' translates to: $w \Rightarrow Pr$

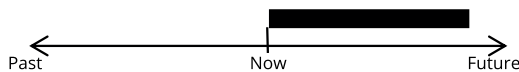
[Fig. 2a] Expressing tensed notions with indeterminate operators.

The determinate operators are H ('has always been') and G ('is always going to be'); with d = 'it is day', n = 'it is night' we can state 'Night has always followed day and day has always followed night': $H((d \Rightarrow Fn) \wedge (n \Rightarrow Fd))$



(Here, black represents n and white d).

In another example, using e = 'planet Earth is present', we can say 'One day, Earth will be irrevocably destroyed': $FG\neg e$



Note that here the F is controlling $\neg e$, i.e. the end of the line.

[Fig. 2b] Determinate tense logic operators.

arose of whether to consider time as a dynamic continuum or a series of static instants. It was given a particularly thorough examination in the work of John McTaggart Ellis McTaggart,⁵ who used this in his provocative argument for the unreality of time (McTaggart 1908).

5 The unusual duplicity in his name is due to him being named John McTaggart Ellis at birth, after his great uncle John McTaggart. When the latter died, he left his estate to the Ellis family on the condition that they changed their family name to McTaggart. They complied by adding it to the end of every family member's name (McDaniel 2020).

System A

Axioms:

p (where p is a tautology)

$G(p \Rightarrow q) \Rightarrow (Gp \Rightarrow Gq)$

$H(p \Rightarrow q) \Rightarrow (Hp \Rightarrow Hq)$

$p \Rightarrow HFp$

$p \Rightarrow GPs$

Using these, the two 'weaker' operators can be defined:

$Pp = \neg H\neg p$

$Fp = \neg G\neg p$

[Fig. 3a] Prior's A system of tense logic.

System B

We introduce a set of instants I and a precedence relation, $<$. A further operator T , taking parameters p for proposition and i for instant, gives:

$T(i, p)$ is true when p is true at i

We can now define the temporal operators using this model (and a further instant j):

$T(i, Fp) = \exists j \cdot i < j \wedge T(j, p)$

$T(i, Pp) = \exists j \cdot j < i \wedge T(j, p)$

G and H can be constructed in a similar way:

$T(i, Gp) = \forall j \cdot i < j \Rightarrow T(j, p)$

$T(i, Hp) = \forall j \cdot j < i \Rightarrow T(j, p)$

System A's axioms are also all axioms of System B.

[Fig. 3b] Prior's B system of tense logic.

McTaggart classified logical systems about time into two types: A and B. The A-theory view of time is that presented above—it is the logic of tenses and is operator-first, with statements made relative to a privileged "now." Operators are defined axiomatically (e.g. $Gp = \neg F\neg p$) and they are able to translate temporal statements lacking measurements into a formal setting. See Figure 3a for definitions.

By contrast, B-theory systems start by defining the notion of a comparable instant. Logical propositions are associated with instants using an operator that lets us judge the truth of a proposition at a given instant. B-theory relies on concepts of simultaneity, before, and after as its main notions; critically, it does not afford privilege to any particular instant, not even "now." The same temporal operators can be defined as in A-theory, but are now given in relation to the series of instants. Definitions, following Prior (1967) and Øhrstrøm and Hasle (2007), are given in Figure 3b.

Operators from A-theory can be defined with relation to the time series of B-theory, as seen in Figure 3b. Øhrstrøm and Hasle (2007, §3.2) detail an argument that B-theory should be seen as the "basic" system, with the premise that B-theory concepts cannot be expressed in A-theory since it has no way to express the ordering of events. That the B-theory is seen as the standard is reflected in its primacy in the Stanford Encyclopedia of Philosophy

entry on temporal logic (Goranko and Rumberg 2022). However, Prior was opposed to that view, in part motivated by his own realist philosophy which privileged human-intuitive models, including (in his view) A-theory. Øhrstrøm and Hasle (2007, §3.2) agree, and provide a mechanism through which A- and B-theory concepts can be freely intertranslated.

It is notable that many modern logics were developed by people familiar with complex mathematics (e.g., Russell, Kripke, Scott). By contrast, Prior did not train in maths until later in life, and tense logic was developed in an almost purely philosophical setting. This created an emphasis, at least initially, on using tense logic for conceptual investigations, hence the strong connection with statements in natural language. Tense logic did, however, still attract the intellectual cachet of formalism, though its form differed: A-theory lent itself to an axiomatic presentation and B-theory to a semantic model (*ibid.*, §2.8). This is how they are presented in Figure 3a and 3b.

The distinction is relevant for the application of tense logic to computing, for which the potential utility of discrete-time B systems was recognized by Prior (1967, 67). He wrote they “are applicable in limited fields of discourse in which we are concerned only with what happens next in a sequence of discrete states, e.g. in the working of a digital computer.” Despite this observation in 1967, the impact of tense logic on computing did not come until well into the next decade. This history is traced in the next section.

4. Time, for Computers

One early glimpse of a logic of time in a computing setting came from John McCarthy and Patrick Hayes (1969), writing a review of ideas from philosophy that could be exploited for their artificial intelligence (AI) work at Stanford. The authors proposed a “fluent” function which allows statements such as “given one situation, a certain other will eventually result.” They explicitly made the connection between this situation calculus and Prior’s tense logic, presenting intertranslation of the relevant operators. They also proposed a “formal literature,” which is “like a formal language with a history: we imagine that up to a certain time a certain sequence of sentences have been said. The literature then determines what sentences may be said next” (*ibid.*, 32). As well as being an interesting anticipation of natural language processing AI in the 2010s, this notion has a reflection in later models of temporal logic that use sequences of (abstract) computer states as models for history, and which determine the future properties of the system or its allowable actions.

While this paper shows glimpses of how the ideas of tense logic could be used in computing, it seems to have made little impact on the early development of temporal logic in computer science. This could be because of its focus on the AI domain, quite different to the applications for which early temporal

logic work were developed (it has a large number of citations in AI literature). Instead, the line of work taking tense logic to concurrency goes via another Stanford connection, the work of Zohar Manna on termination.

The task of verifying the correctness of programs is difficult and has a long history (Jones 2003). An early proponent was Bob Floyd (1967), who drew flow-charts of programs with logical assertions expressing properties at particular points. Floyd then conducted semi-formal proofs using axioms about the program behavior and input to trace through its intended execution, demonstrating that particular conditions about the program output would be true. In this way, Floyd, and others who followed in his style such as Hoare (1969), argued that a program could be verified correct.⁶

However, should there be some error in a program or input that prevents it terminating (for example, a loop which is supposed to count down to a target value instead counts upward indefinitely), the program's correctness cannot be shown. Since most programs in the 1960s were "functional" (i.e., intended to produce an answer), proving termination was a desirable feature. Due to the halting problem (Davis 1965), an algorithm to automatically determine whether an arbitrary program terminates is impossible, but for some programs it is possible to construct a proof of termination. Floyd used his annotations to show that every step of the program decreased some quantity towards a limit (e.g., a loop counter decrementing to zero). The inherent future tense in the question "Will this program terminate?" provides a hint to why this line of work eventually involved tense logic.

We see in the work of Floyd and Hoare some early attempts to bring programs into the domain of mathematical proofs. There is yet to be any explicit connection with logic (though later Hoare's work would be termed "Hoare logic") and neither author provides citations to literature outside the computing field. Instead we can observe that their use of logicomathematical constructs like propositions and predicates assumes the audience has familiarity with these concepts already. From the framing, the authors position themselves within the tradition of formal semantics for programming languages, one of the earliest academic spheres of theoretical computer science (Astarte 2022). By the late 1960s, the precedent for these approaches was well established, though much criticized; indeed, this new strand of program verification was in some respects an answer to the difficulties of formal semantics (*ibid.*). The application of techniques explicitly influenced by logic is yet to appear, and, as we will see, was largely reinvented.

6 Later proponents of program verification would come to talk (more precisely) about a program "meeting its specification" and detractors argued about the applicability of proofs made about abstract representations to the physical working of programs—but these concerns are out of scope for the current chapter and the interested reader is directed to MacKenzie (2001, Chapter 6).

The thread begins properly with Floyd's student Manna (1939–2018), who studied mathematics at the Technion in Haifa, Israel, before embarking on a PhD at Carnegie Mellon. His dissertation "Termination of algorithms" (Manna 1968) took Floyd's ideas further and included significantly more logical weight. The idea was to translate an algorithm (an already abstract representation of a program's functionality, not dissimilar to Floyd's flowcharts although less pictorial) into a series of statements of first-order predicate calculus. Through a careful manipulation of these predicates, now in the space of logic, a particular statement could be formulated, the (logical) satisfiability of which would indicate the termination of the original algorithm.⁷ Through a series of papers, Manna (1969a, 1969b) showed that the translation into logical predicates could be achieved for more concrete programs. This decision to move away from merely adding logical statements to programs (as Floyd had done) and instead to translate wholesale into predicate calculus represented a new way of thinking about programs in a logical way—forefronting the logic as the space for reasoning.

Manna (1970) next became interested in non-deterministic programs—those which, rather than being functional, have a number of equally correct possible terminations. This work led ultimately towards his collaboration with another Stanford postdoc, Ed Ashcroft (Ashcroft and Manna 1971), which addressed (inherently non-deterministic) concurrent programs. However, this approach to concurrency was ultimately discarded by Manna and did not inform his further work; for more on this, see Jones (2023).

Instead, Manna (and Pnueli) came to favor a modal-logic based approach, which appeared in a paper by Burstall (1974). Burstall had encountered Manna's approach to program correctness, but found his wholesale translation into predicate calculus overly complicated for problems that Burstall felt were relatively simple. Instead, he favored a more informal proof of program correctness and termination achieved by "stepping through" a pen-and-paper simulation of a program's execution. This proof was built using predicate assertions attached to points in the program.

The key distinction came from Burstall using two kinds of assertion attachment. In the standard Floyd approach, as used by Manna, an assertion must always be true every time the control flow reaches that point in the program. Burstall, however, introduced another kind of assertion which stated that execution of the program would eventually reach that point when a particular assertion was true (*ibid.*, 308). To illustrate the distinction, a typical Floydian assertion within a loop whose termination criterion is $i > n$ might state $i \leq n$;

7 Satisfiability is a property of formulae in mathematical logic: a formula is satisfiable if there exists an value for each of its variables which results in the overall formula being true. For example, $i < 0$ is satisfiable when i can range over the integers, but not when i is confined to the natural numbers.

Burstall's assertion would state $i > n$. The distinction might seem rather trivial at this point, and, indeed, Burstall shows that there is not much expressive difference in his approach and Floyd's, since in functional programs one would expect control to pass through every point in a (well-written, terminating) program. Instead, some proofs come out easier with Floyd's and some with Burstall's. The real value would come later, with an application to non-deterministic and concurrent programs, where the multiple valid outcomes meant that control would not be guaranteed to flow through a whole program.

Burstall made the important observation in the paper's conclusion that what he wrote represented a simple form of modal logic. His "eventually" notion was like the modal possibility operator and a standard Floyd-like assertion was like the necessity operator. The possible world semantics of modal logic establishes an interpretation for modal statements: a proposition of the form "possibly p " is true if a world in which p is true is somehow accessible from the actual world. Burstall's formulation used "possible states" and their accessibility was shown through the path of program execution. Clearly Burstall was deeply familiar with logic literature since he could precisely state to which system of Hughes and Cresswell (1968) this corresponded (S5)—however, despite his accessibility notion connecting states of computation through time, he did not make the connection to Prior's tense logic. The fact that this section comes at the conclusion, and is not used as a motivator throughout the text, suggests that Burstall did not consider this the major contribution of his work—perhaps seeing it instead as something of a curiosity rather than a significant tool for the programmer to use.

Burstall's ideas got back to Manna, and provided the inspiration for a new direction in his work, reported on in a joint paper with Richard Waldinger (1976). Now the concept of temporality appears at last in a computer science context, with the title reading "Is 'sometime' sometimes better than 'always'? Intermittent assertions in proving program correctness." Despite this there are no references to the logic literature—the authors do not follow Burstall's lead. Instead the paper is situated firmly in the program correctness paradigm. "Intermittent assertions," a term coined in this paper as a way to describe Burstall's idea, and opposed to the Floyd-style assertions that they called "invariant," are used to prove the correctness and termination of programs in a single proof. This is seen as the major reason to prefer intermittent assertions, though the authors do note in the conclusion that while termination is an interesting property, many useful classes of program do not terminate, and for those, intermittent assertions might also turn out to be appropriate. This line of work was followed by another Israeli, Amir Pnueli.

Pnueli (1941–2009), according to his long-time collaborator David Harel (2010), was a quiet and gentle man, generous; his habit of running late with work did not prevent his contributions being recognized for the 1996 Turing Award

(Zuck 2019). Pnueli had a background working on applied mathematics before moving to Stanford as a postdoc in 1967 (ibid.). There, he worked with Manna and knew his early ideas; their work together concerned moving problems from computing and programming into first-order predicate logic, but there was yet to be any sign of modal or tense logic.

By the mid-1970s, Pnueli had returned to Israel, setting up a new computer science department at Tel Aviv University, where he became interested in non-terminating and cyclic programs, such as operating systems and—critically—concurrent programs (Francez and Pnueli 1978). Viewing such programs as functional transformations of input into output worked rather poorly. Pnueli did not wish to give up on logic, but came to believe that a dynamic approach was needed. He expressed this in a speech in 2000, after receiving the Israel Prize (quoted in Zuck 2019):

In mathematics, logic is static. It deals with connections among entities that exist in the same time frame. When one designs a dynamic computer system that has to react to ever changing conditions, ... one cannot design the system based on a static view. It is necessary to characterize and describe dynamic behaviours that connect entities, events, and reactions at different time points. Temporal Logic deals therefore with a dynamic view of the world that evolves over time.

The early work of Francez and Pnueli did not yet use the language of temporality nor did they cite Manna and Waldinger (1976), suggesting their work happened somewhat independently. Francez and Pnueli analyzed the ongoing behavior of a program as its key property, and wanted to formalize how a program reacted appropriately to stimuli. This lent itself to statements in the form “if input x happens, then at some point the program will do y in response.” By introducing an explicit time variable to propositions about their computational statements, these inherently temporal statements could be expressed. Over these discrete, time-ordered sequences of states, an operator Ev [eventually] was used. This is a clear example of a B-theory system of tense logic, though it was reinvented in this paper. While Pnueli and Francez knew of Burstall's work and followed a similar strategy for their proofs, they did not take up his suggestion for using modal logic.

One computer scientist who did present his work with a strong grounding in logic was Fred Kröger, who was working at Technische Universität München (TUM) in the mid 1970s. Inspired by Burstall, but unaware of Manna and Pnueli, his program logic is likely the earliest example of tense logic in computing. Kröger (1975) argued that since algorithms are dynamic, a logical system is needed to cope with propositions that become true or false over time. Kröger introduced a language for program logic that had specialized operators to

represent specific program components such as branching and looping, which he formalized using Kripke semantics, a standard method for modal logic.

A few years later, Kröger (1978) became aware of Manna and Waldinger's work and showed how their approach connected to the literature of logic. Here, he presented a complete logical system with operators *nex* and *som*. While we have seen the latter, an F equivalent, emerge many times (and implicitly seen G appear in the guise of Floyd-style invariant assertions), this is the first discussion of a "next" operator in computing. Only applicable to discrete-time systems, this operator indicates that a particular proposition should hold in the state directly succeeding the one in which the term is evaluated. Not typically included in Prior's tense logic, a chapter on "Non-standard Logics" in his later book considered this operator (Prior 1967, § IV.3), which Prior called T (for "tomorrow," and its complement Y for "yesterday") and that was due to Dana Scott, who had some widely cited work in this area that was never published (Copeland 2020).

In response to Kröger's presentation, Pnueli mentioned that he was working on a similar system for concurrency and non-determinism using only the "sometimes" and "always" modalities. In the years since his paper with Francez, Pnueli had encountered these modal notions. While on sabbatical at the University of Pennsylvania, Pnueli was working on problems stemming from his partnership with Manna, according to Øhrstrøm and Hasle (2007, 344) who had personal communication with Pnueli. At Penn, Saul Gorn showed Pnueli *Logic of Commands* (Rescher 1966), which was not directly useful, but provided a reference to a book by Nicholas Rescher and Alasdair Urquhart (1971) called *Temporal Logic*, which did inform his work. Pnueli said this was "late 1975 or early 1976"; given that it was not mentioned in the paper with Francez, later seems more likely. With this material in hand, Pnueli (1977) was able to make a proper link to the logical literature. Here, he synthesized ideas from Burstall, Manna, and Kröger with some firm modal logic, calling the result "temporal logic"—noting also that McCarthy and Hayes (1969) had made some suggestions along these lines. Intermittent assertions could now be formalized using "temporal reasoning" and applied to concurrency. In a later publication expanding the idea Pnueli (1979) uses state sequence semantics and includes a Kröger-style "next" operator X alongside F and G, now explicitly invoking the connection with Prior.

Here we can see a new phase in the application of logic to computing. While Burstall, Manna, and Pnueli (and their collaborators) had taken ideas here and there from logic, much of their work in the 1970s created new ways of thinking about programming largely ignorant of the related literature in logic. By the end of the decade, the links were discovered, and a new basis for the computing logics could be presented, as part of a distinguished lineage. In contrast, Kröger explicitly evoked much of the logic canon in his work, but

perhaps because of its lack of immediate connection to the program-proving paradigm, it was somewhat overlooked—though all the same components were present as in Pnueli’s paper a few years later, it was Pnueli that received the Turing Award. This will be discussed further in the current chapter’s conclusion, but first, let us examine the manner in which Pnueli’s temporal logic was presented with its new logic grounding.

5. Temporal Logic Canon

By 1980, Manna held positions at both Weizmann and Stanford and Pnueli was at Weizmann; the two built a network of researchers into temporal logic located around Israel. Many of these can be seen as collaborators during this time and in the acknowledgments of a series of joint works by Manna and Pnueli (1981a, 1981b, 1982, 1983). These papers lay out a canonical form for temporal logic, specifically positioned as a method for abstract reasoning about, and verification of, concurrent programs.

By now, Manna and Pnueli had become versed in modal and temporal logic, and reinterpret their previous work in this grounding—making reference to Prior (1967) and Rescher and Urquhart (1971). Indeed they build their own logical system by presenting modal logic as a well-known natural starting point. They provide the justification for employing modal logic as a basis for temporal logic by explaining that it fixes one variable and allows variation over others. While standard predicates and quantifiers allow expression of properties of a particular program state, temporal operators indicate relationships between states. Using modal logic is not wholly necessary for a temporal logic system, since Pnueli had already shown that time could simply appear as a parameter in predicate logic systems, but by making time the major variable its importance as the factor linking states is more clearly emphasized.

The logic system uses a discrete sequence of ordered states and operators are derived from this—in other words, a B-system. The technical details are represented in Figure 4; note the use of pictorial operators borrowed from modal logic rather than the uppercase letters Prior used. The system explicitly includes an axiom (no. 39) of “forwards linearity”: there is no branching future here.

Having built a temporal logic system that is so far entirely abstract—applicable to any kind of tensed statement about events—Manna and Pnueli work on the parts specific to programs. By deriving axioms from particular programs and adding them to the overall temporal logic, expressions of properties of those programs can be made. The proof principles and decision procedure of the logical system can then be used in order to prove them. Various proof strategies are presented, which build on the intermittent assertions approach of Manna and Waldinger (1978) as well as Manna’s early work on termination.

A temporal logic universe is a collection of states and an accessibility relation R . A state s has precisely one accessible next state s' defined as its discrete time successor via the function ρ so $\rho(s, s')$ holds. The transitive and reflexive closure of ρ , R , is the accessibility relation. $R(s, s')$ is true when s' is in the future of, or is identical to, s .

The model created by this universe is an infinite sequence of states:

$$\sigma = s_0, s_1, s_2, \dots$$

and

$$\forall i \geq 0, j \geq i \cdot \rho(s_i, s_{i+1}) \wedge R(s_i, s_j)$$

Notably, the index updates as progress is made: s_0 is always the current state, and s_1 the next state.

Using $s(w)$ as the truth value of the formula w in state s , the temporal operators are defined as follows:

$$s_0(\Box w) = \forall i \geq 0 \cdot s_i(w)$$

$$s_0(\Diamond w) = \exists i \geq 0 \cdot s_i(w)$$

$$s_0(\circ w) = \exists i \geq 0 \cdot s_i(w)$$

[Fig. 4] Temporal operators defined in Manna/Pnueli style

Pnueli and Manna provide an intriguing advantage for using this latter system: less temporal reasoning is required. The implication here is that the additional complexity of reasoning using the logical system might be off-putting to potential users.

The presentation of temporal logic in this series of papers, which forms the basis of textbooks written by Kröger (1987) and later Manna and Pnueli themselves (1992), paints a different picture to that which is outlined in the current chapter. This temporal logic system, which was built up by a variety of authors over a series of years and started with systems that worked very closely with programs—and that might conceivably form part of an advanced programmer's workflow—is presented here firmly as a variety of the well-established modal logic. This reframing of their work as part of a time-honored abstract logical world mirrors the “reflective closure” of the formalization of programming language semantics, a related field of computer science (Astarte 2022), and is typical of the way computers were reinvented as logic machines in the 1980s (Priestley 2011).

The four Manna/Pnueli papers discussed in this section indicate the maturing of temporal logic into a central canon, with the duo at the center and support from their growing research environment. The topic started to be taught, for example by Kröger at TUM from 1983, out of which course he wrote a textbook combining the core ideas from Manna and Pnueli with some aspects of his own earlier logic (Kröger 1987). Work throughout the 1980s on temporal logic continued in two veins: experimentation with the core logical system—adding operators, testing expressiveness, decision procedures, and so on—this activity often coming from the Israeli group; and applying the ideas from temporal logic to practical tools and approaches, often externally to Manna and Pnueli's team. Those applications are explored in the following section.

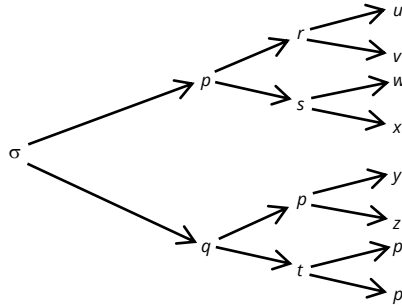
6. Problems of the Past, Logics of the Future

With an established canon of temporal logic to examine, we can now return to some of the old logical questions considered previously in this chapter: the fundamental basis for a system, and the branching or linear nature of time. As explored earlier, McTaggart identified two basic conceptions for thinking about tensed statements, A- and B-theory, which Prior subsequently considered in some detail. Every computer science publication cited in the present chapter uses only a B-theory basis: Why? One reason is the ostensible focus on programming that set state sequences as the domain of discourse, thanks to an established literature on the syntax and semantics of programming languages being heavily state-based (see Astarte 2019, 2022). Prior himself had noted that B-theory could be applicable to problems involving computers. It is however curious to wonder why the canonical series of papers, which uses the language of abstract logical systems so clearly, does not even explore an operator-first A-system. Another potential explanation is that insignificant linkage to practical programming was a good way to get computing theory dismissed as irrelevant in the 1960s and 70s—even though that link was rarely exercised (Astarte 2022). The temporal logic canon discussed above does give example “programs” for treatment within the logical framework, but these are all rather small, and none are actually written in a real programming language (rather, pseudocode is used).

Leslie Lamport (1941–) was interested in providing a more practical framework for this kind of logic, and, indeed, also in the question of branching time. Lamport studied a PhD in mathematics, and, unusually for a someone who made significant contributions to computer science theory and practice—even winning the Turing Award in 2013—never had a permanent academic position and spent his career in industry. Lamport became interested in concurrency through algorithms; he saw a paper on the mutual exclusion problem in *Communications of the ACM* and thought he could write a better solution, eventually developing the “bakery algorithm” (Lamport 1974). In the process, he came to believe that concurrent algorithms need reliable proofs (Hoare and Lamport 2020).

Like many others mentioned in this chapter, Lamport (1977) invented his own temporal system, in his first publication on proving properties of concurrent programs. It hinges on his operator $A \rightsquigarrow B$, which he reads as “if a legitimate execution reaches a state in which A is true, then it will subsequently reach a state in which B is true.” Like with many other authors, Lamport appears to have invented this independently of the logic literature, including no citations thereto, and though he references Manna’s work, it is the early material with Ashcroft that has no temporal aspects.

Consider a branching future as illustrated below; the current state (used for interpretation) is σ .



In Lamport's view, 'sometime p ' is true, since every possible future path has p true at some point. However, while 'sometime z ' is true, since one future path has z true, 'not always not z ', ($\neg\Box\neg z$ or $\Diamond z$) is *not* true, since there are branches in which z is never true. In the style of Ben-Ari, Pnueli, and Manna (1983), $\forall Fp$ and $\exists Fp$ are both true; $\exists Fz$ is true while $\forall Fz$ is false.

[Fig. 5] Different interpretations of future tensed propositions in a branching future

By the end of the decade, Lamport (1980) had become aware of both temporal and tense logic, providing many citations to Rescher and Urquart (1971). Lamport's concern was now that linear time might not always be the best model, as reflected in the title "'Sometime' is sometimes 'Not never'"—an obvious reference to Manna and Waldinger (1976). He writes out a branching time temporal logic, noting that branching and linear time are equally but not equivalently expressive. In branching time, in which every possible future is equally real, "sometime p " should denote that p is true at some point in every future path. However $\neg\Box\neg p$ (or $\Diamond p$) means that it is not true that every future path has a continual false value for p ; i.e., there is at least one future path in which p is true at some point. See Figure 5 for an illustration.

Lamport argues that branching time temporal logic is better for modeling non-determinism, in the sense of automata theory, where a non-deterministic machine is one that pursues all its possible courses simultaneously and terminates successfully when one course succeeds.⁸ On the other hand, concurrency is often modeled by a system in which actions of concurrent agents happen sequentially, but the order of which is non-deterministic. Lamport made the case that reasoning about concurrent programs is concerned only with the sequence of events that actually happens in the future of the program, and so linear time temporal logic is more suitable since it constrains the future to one path.

8 This kind of automata-theoretic approach is used in the work of Pratt (1976, 1979) and Harel and Pratt (1978).

Branching time was addressed by the Israeli group: Ben-Ari, Pnueli, and Manna (1983) present some ideas in this respect. To set up branching time, the semantic model is changed from state sequences to state trees, rather like in Figure 5. The future is then a tree rooted in the current state; quantifiers range over branches and temporal operators within branches. This means that there is a distinction between the operator $\exists F$ (true at some point in the future in at least one branch) and $\forall F$ (true at some point in the future on every branch).

This allowed the expression of yet more properties, and, the authors claim, presentation of non-deterministic programs more naturally—yet some aspects remained inexpressible in either given system. Branching time, generally, is less suited to discussions of fairness,⁹ since the tree of all possible computations includes unfair paths; yet for discussion of non-deterministic properties, it is the only model to use.

Lamport frequently wrote about temporal logic in a rather critical tone—in particular he felt many systems were “too expressive,” especially when they included “next” operators (Lamport 1980). In developing a system for specifying programs using temporal logic, Lamport wrote about needing, for example, “a lot of ordinary mathematics glued together with a little bit of temporal logic” (Lamport 1999), presumably in a backhanded reference to the large quantities of complicated mathematics in the Israeli group’s papers. Lamport’s system TLA (Lamport 1994), and its successor TLA+ (Lamport 1999), was intended for practical use by software engineers for specifying concurrent systems. The idea was that a programmer could work out the core functionality of their proposed system using a TLA model and verify some important properties before the complexities of implementation made the task more difficult. He emphasized a “compositional specification” approach with the basic module unit used to build more complex specifications. The underlying temporal logic system appears simple and uses $\Box$ as its only operator, though assertions are allowed over pairs of states, which he calls “actions” and that implicitly incorporate a notion of “next.” While these actions are inherently temporal, the reasoning about them can be non-temporal, simplifying the process. Following his earlier remarks about its suitability for concurrent systems, TLA uses linear time logic. TLA+ did indeed end up seeing some success in industrial use, notably by Amazon Web Services (Cook 2018; Newcombe et al. 2015), though both utilized additional tools to check the specifications—specifically, the model checker TLC (Yu, Manolios, and Lamport 1999).

The model-checking paradigm also has its roots in temporal logic, though it has grown to be a significant tool of the verification community. While

9 This key property of concurrent systems, which essentially states that no concurrent agent that is able to act should be prevented indefinitely from doing so, is treated by another collection of Israeli researchers (Gabbay et al. 1980).

Lamport downplayed the mathematics and temporality in TLA, Clarke and Emerson (1981) were not afraid to dive into the technicalities of branching time logic. According to Emerson (2010), he had been trying to work with concurrency in a Hoare logic framework and found the manual proof constructions difficult and time-consuming. Emerson heard Manna give a talk on fixed points at the University of Texas in 1975, and this led to a team at Texas trying to apply this to parallel programs; the results were presented by Emerson and Clarke (1980). Working in this setting, the duo realized their “fixed point” semantics corresponded to the branching temporal logic of Ben-Ari, Pnueli, and Manna (1983). They felt this branching logic struck a nice balance between expressiveness and decidability, though the complexity of many concurrent programs still caused difficulties. Clarke and Emerson (1981) therefore used a “synchronization skeleton”: the program’s concurrency management aspects with all the sequential parts hidden.

The model-checking approach starts with a specification written in the CTL language. From this, a synchronization skeleton can be automatically synthesized and will have a finite number of states as a consequence, making it amenable to mechanized checking. Their branching time logic is used since the ability to quantify over paths of computation, not available in linear time logics, is useful in program synthesis. Properties can be proven about the skeleton and once the synchronization is proven correct the rest of the program can be written. The program synthesis is based on the decision procedure for the satisfiability of CTL; Clarke and Emerson (1981) note that although the procedure is potentially exponential, the skeletons tend to be quite small.

This work on the branching time logic CTL, and the surrounding development and verification ideas, led to the Turing Award being granted to Emerson and Clarke in 2007, alongside Joseph Sifakis, who developed essentially the same idea independently (Emerson 2008). The paradigm is called “model checking” since their algorithm checks whether the synchronization skeleton is a proper model of the properties desired for the program. Since the 1980s, there have been many developments in the area, seeing particular success in applications to hardware verification, an area where problems tend to be smaller in size and more regular (Emerson 2010). As well as the Amazon examples discussed above, other industrial uses were presented by Emerson and Namjoshi (1998), who also somewhat poetically claimed that model checking addressed Gottfried Wilhelm Leibniz’s dreams of universal calculus.

Like TLA+, model checking in the early 2000s represented a more prosaic dream: the long-awaited industrial applicability of theoretical computer science and the formal methods community. Emerson (2008) reports Daniel Jackson claiming model checking “saved the reputation” of formal methods, and, grudgingly, Edsger Dijkstra admitted it was an “acceptable crutch.”

From its early recreations in the 1970s, and ancient roots in philosophy and theology, the logic of time has, at least, found a place in the 21st century.

7. Conclusions

Though the origins of the logic of time have ancient and philosophical roots, concerns about effective ways to express temporal concepts continue to resonate even when placed into an entirely different context, that of digital computers. Whether attempting to investigate the tensed nature of human language, or to state critical properties of concurrent programs, the precision of a logic of time provides a useful tool. The ability to state difficult tensed notions formally yet intuitively was as essential for Prior trying to represent the human experience as for Lamport attracting software engineers to a specific paradigm. It is perhaps less surprising that philosophy should prove relevant to concurrent programming when we recognize the world as a giant concurrent system, full of independent agents behaving unpredictably.

In the computing context the question of time's structure as linear or branching changes from a theological conundrum pitting omniscience against free will to one of the relative merits of non-determinism or fairness in concurrency. This choice presents real technical challenges over which statements can be phrased in a logic system and the significant difficulties of combining them. Choosing a structure for the future affects what can be proven to be true: the way the world is viewed, is, in a sense, changing its very nature. Despite this, the two practical systems for temporal logic discussed here, model checking and TLA, effectively manage this choice and the TLC system even uses aspects of both.

By contrast, the debate over A- and B-theory as the fundamental basis for a logic of time is almost entirely ignored within computing. It is simply taken as automatic that a B-system based on state sequences is the appropriate choice, likely an extension of existing views on the semantics of programming languages and proofs of programs. Perhaps this also stems from the lack of engagement by the early computer science work with the long history of time in logic. Unlike Peirce, Łukasiewicz, and Prior, all of whom studied and used historical logic, temporality took a circuitous route into computing.

Though Prior had already identified the potential application of tense logic to computers in 1967, and McCarthy and Hayes noted some connections, most computer scientists did not engage with the logic literature until the early 1980s. Instead, similar ideas were slowly reinvented and later rationalized when the connections were realized. It is interesting to observe that the exception to this rule is Kröger, whose work connected to the logic literature immediately; despite—or perhaps even because of—this, his work did not

cause a great deal of impact and he is often overlooked in the internal history of the field.

Once a temporal logic system research agenda was established, largely around Manna and Pnueli, experimentation with the system formed the dominant trend of the 1980s. The establishment of the agenda can be seen in the way that many new introductions began with student projects and were then incorporated into the canon via joint papers with Manna and/or Pnueli—e.g., branching time with Ben-Ari (1983).

Penetration of these ideas into practical applications took a long time, and it is worth observing that the two discussed here took different routes: simplicity in TLA to encourage programmers to specify with it; and complexity backed up with tool support for model checking. It is interesting to observe that Manna and Pnueli, as well as Lamport, find the complexity of temporal reasoning sufficiently off-putting as to sell ways to avoid it as positives in some of their systems. Further, both practical temporal systems mentioned here, TLA and CTL, use a simplified program structure for temporal reasoning to cut down on the complexity and enhance the usability of their systems.

The standard narrative in computer science that computers grew from the application of logic theory is lacking in nuance, as Priestley (2011) notes; rather, much of theoretical computer science stems from programmers inventing “new” tools and discovering only later they can be legitimized and codified by connecting them with mathematical or logical practice. The choice of symbols for temporal operators reflects this: starting with program-like keywords such as “eventually” or “sometime,” Prior’s single capital letters—though not his taste for full Polish notation—were briefly in vogue until the connections were made with modal logic, and those symbols taken over, representing the assumption of a legitimate logical appearance. The temporal logic research program in the 1970s and 80s is part of the re-examination of computing objects (physical and abstract) in the rediscovery of logic, a rationalizing of existing practice for concurrent programming, as had happened for formal semantics in the 1960s (Astarte 2022). The assumption of this intellectual pedigree may be part of the reason why temporal logic is a particularly prestigious field of computer science—this chapter features four Turing Award winners. Temporal logic, then, is an example not of a new branch of logic or even a practice with direct applicability to programming, but a combined object between the two, the re-situation of old and ancient ideas in a new context.

The research leading to this chapter received funding from Leverhulme Trust Grant No. RPG-2019-020. Many thanks to attendees at ICHST 2021, Arianna Borrelli, Marie-José Durand-Richard, Cliff Jones, Mark Priestley, and Markus Roggenbach for providing useful comments and suggestions.

For the purpose of Open Access, the author has applied a CC BY licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

- Ashcroft, Ed A., and Zohar Manna. 1971. "Formalization of Properties of Parallel Programs." In *Machine Intelligence 6*, edited by B. Meltzer and D. Michie, 17–41. Edinburgh: Edinburgh University Press.
- Astarte, Troy Kaighin. 2019. "Formalising Meaning: A History of Programming Language Semantics." PhD thesis, Newcastle University, Newcastle, UK.
- . 2022. "'Difficult Things Are Difficult to Describe': The Role of Formal Semantics in European Computer Science, 1960–1980." In *Abstractions and Embodiments: New Histories of Computing and Society*, edited by Janet Abbate and Stephanie Dick, 126–45. Baltimore: Johns Hopkins University Press.
- . 2023. "From Monitors to Monitors: An Early History of Concurrency Primitives." *Minds and Machines* 34: 51–71. <https://doi.org/10.1007/s11023-023-09632-2>.
- Ben-Ari, Mordechai, Amir Pnueli, and Zohar Manna. 1983. "The Temporal Logic of Branching Time." *Acta Informatica* 20, no. 3: 207–26.
- Burstall, Ron M. 1974. "Program Proving as Hand Simulation with a Little Induction." In *Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5–10, 1974*, edited by J. L. Rosenfeld, 308–12. Amsterdam: North-Holland.
- Clarke, Edmund M., and E. Allen Emerson. 1981. "Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic." In *Workshop on Logic of Programs*, edited by D. Kozen, 131: 52–71. Berlin: Springer-Verlag.
- Cook, Byron. 2018. "Formal Reasoning About the Security of Amazon Web Services." In *Computer Aided Verification: CAV 2018*, edited by G. Weissenbacher and H. Chockler, 10981, 38–47. Cham: Springer.
- Copeland, B. Jack. 2020. "Arthur Prior." In *The Stanford Encyclopedia of Philosophy (Spring 2020 Edition)*, edited by Edward N. Zalta. Stanford University. Accessed November 2024. <https://plato.stanford.edu/archives/spr2020/entries/prior>.
- Davis, Martin. 1965. *Computability and Undecidability*. Mineola: Dover.
- Dunning, David E. 2020. "Writing the Rules of Reason: Notations in Mathematical Logic, 1847–1937." PhD thesis, Princeton University, Princeton, NJ.
- Durand-Richard, Marie-José. 2023. "Boole's Symbolized Laws of Thought Facing Empiricism." In *Logic in Question: Talks from the Annual Sorbonne Logic Workshop (2011–2019)*, 97–118. Cham: Springer.
- Emerson, E. Allen. 2008. "The Beginning of Model Checking: A Personal Perspective." In *25 Years of Model Checking*, 27–45. Cham: Springer.
- . 2010. "Meanings of Model Checking." In *Concurrency, Compositionality, and Correctness: Essays in Honor of Willem-Paul de Roever*, edited by Dennis Dams, Ulrich Hannemann, and Martin Steffen, 237–49. Berlin: Springer.
- Emerson, E. Allen, and Edmund M. Clarke. 1980. "Characterizing Correctness Properties of Parallel Programs Using Fixpoints." In *International Colloquium on Automata, Languages, and Programming*, 169–81. Berlin: Springer.
- Emerson, E. Allen, and Kedar S. Namjoshi. 1998. "Verification of a Parameterized Bus Arbitration Protocol." In *International Conference on Computer Aided Verification*, 452–63. Berlin: Springer.
- Floyd, Robert W. 1967. "Assigning Meanings to Programs." In *Mathematical Aspects of Computer Science. Volume 19: Proceedings of Symposia in Applied Mathematics*, edited by Jacob T. Schwartz: 19–32.. Providence: American Mathematical Society.
- Francez, N., and A. Pnueli. 1978. "A Proof Method for Cyclic Programs." *Acta Informatica* 9: 133–57. <https://doi.org/https://doi.org/10.1007/BF00289074>.

- Gabbay, Dov, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. 1980. "On the Temporal Analysis of Fairness." In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 163–73. New York: Association for Computing Machinery.
- Goldblatt, Robert. 2006. "Mathematical Modal Logic: A View of Its Evolution." In *Handbook of the History of Logic 7: Logic and the Modalities in the Twentieth Century*, edited by Dov M. Gabbay and John Woods, 1–98. Amsterdam: Elsevier.
- Goranko, Valentin, and Antje Rumberg. 2022. "Temporal Logic." In *The Stanford Encyclopedia of Philosophy (Summer 2022 Edition)*, edited by Edward N. Zalta. Stanford University. Accessed November 2024. <https://plato.stanford.edu/archives/sum2022/entries/logic-temporal>.
- Harel, David. 2010. "Amir Pnueli. A Gentle Giant: Lord of the ϕ 's and the ψ 's." *Formal Aspects of Computing* 22, no. 6: 663–65.
- Harel, David, and Vaughan R. Pratt. 1978. "Nondeterminism in Logics of Programs." In *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 203–13. New York: Association for Computing Machinery.
- Hoare, Charles Antony Richard. 1969. "An Axiomatic Basis for Computer Programming." *Communications of the ACM* 12, no. 10: 576–80.
- Hoare, Charles Antony Richard, and Leslie Lamport. 2020. "Virtual HLF 2020 – Dialogue: Sir C. Antony R. Hoare/Leslie Lamport." Heidelberg Laureate Forum YouTube Channel. Accessed November 2024. <https://www.youtube.com/watch?v=wQbFkAkThGk>.
- Hughes, George E., and Maxwell J. Cresswell. 1968. *An Introduction to Modal Logic*. London: Methuen.
- Jakobsen, David. 2020. "A.N. Prior and 'the Nature of Logic'." *History and Philosophy of Logic* 41, no. 1: 71–81. <https://doi.org/10.1080/01445340.2019.1605479>.
- Jones, Cliff B. 2003. "The Early Search for Tractable Ways of Reasoning About Programs." *IEEE Annals of the History of Computing* 25, no. 2: 26–49. <https://doi.org/10.1109/MAHC.2003.1203057>.
- . 2023. "Three Early Formal Approaches to the Verification of Concurrent Programs." *Minds and Machines* 34: 73–92. <https://doi.org/10.1007/s11023-023-09621-5>.
- Kröger, Fred. 1975. "Formalization of Algorithmic Reasoning." In *International Symposium on Mathematical Foundations of Computer Science*, 287–93. Berlin: Springer.
- . 1978. "A Uniform Logical Basis for the Description, Specification and Verification of Programs." In *Proceedings, IFIP Working Conference on Formal Description of Programming Concepts, St. Andrews, NB, Canada, August 1–5, 1977*, edited by Erich J. Neuhold, 441–57. Amsterdam: North-Holland.
- . 1987. *Temporal Logic of Programs. EATCS Monographs on Theoretical Computer Science*. Berlin: Springer.
- Lamport, Leslie. 1974. "A New Solution of Dijkstra's Concurrent Programming Problem." *Communications of the ACM* 17, no. 8: 453–55. <https://doi.org/10.1145/361082.361093>.
- . 1977. "Proving the Correctness of Multiprocess Programs." *IEEE Transactions on Software Engineering* 3, no. 2: 125–43. <https://doi.org/10.1109/TSE.1977.229904>.
- . 1980. "'Sometime' is Sometimes 'Not never': On the Temporal Logic of Programs." In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 174–85. New York: Association for Computing Machinery. <https://doi.org/10.1145/567446.567463>.
- . 1994. "The Temporal Logic of Actions." *ACM Transactions on Programming Languages and Systems* 16, no. 3: 872–923. <https://doi.org/10.1145/177492.177726>.
- . 1999. "Specifying Concurrent Systems with TLA+." In *Calculational System Design*, edited by Manfred Broy and Ralf Steinbrüggen, 183–247. Amsterdam: IOS Press.
- MacKenzie, Donald. 2001. *Mechanizing Proof: Computing, Risk, and Trust*. Cambridge, MA: MIT Press.
- Manna, Zohar. 1968. "Termination of Algorithms." PhD thesis, Carnegie-Mellon University. ProQuest. Accessed November 2024. <https://www.proquest.com/openview/da62febaf87453c725708beec9bbc789/1?pq-origsite=gscholar&cbl=18750&diss=y>.

- . 1969a. "Properties of Programs and the First-Order Predicate Calculus." *Journal of the ACM* 16, no. 2: 244–55. <https://doi.org/10.1145/321510.321516>.
- . 1969b. "The Correctness of Programs." *Journal of Computer and System Sciences* 3, no. 2: 119–27. [https://doi.org/10.1016/S0022-0000\(69\)80009-7](https://doi.org/10.1016/S0022-0000(69)80009-7).
- . 1970. "The Correctness of Nondeterministic Programs." *Artificial Intelligence* 1, nos 1–2: 1–26. [https://doi.org/10.1016/0004-3702\(70\)90002-0](https://doi.org/10.1016/0004-3702(70)90002-0).
- Manna, Zohar, and Amir Pnueli. 1981a. "Verification of Concurrent Programs: The Temporal Framework." In *The Correctness Problem in Computer Science*, edited by R. S. Boyer and J. S. Moore, 215–73. New York: Academic Press.
- . 1981b. "Verification of Concurrent Programs: Temporal Proof Principles." In *Logic of Programs, Workshop*, edited by Dexter Kozen, 131:200–252. Lecture Notes in Computer Science. Berlin: Springer.
- . 1982. "Verification of Concurrent Programs: Proving Eventualities by Well-Founded Ranking." Technical report STAN-CS-82-915. Stanford University Computer Science Department. Accessed November 2024. <http://i.stanford.edu/pub/cstr/reports/cs/tr/82/915/CS-TR-82-915.pdf>.
- . 1983. "Verification of Concurrent Programs: A Temporal Proof System." STAN-CS-83-967. Department of Computer Science, Stanford University. ACM Digital Library. Accessed November 2024. <https://dl.acm.org/doi/book/10.5555/892296>.
- . 1992. *Temporal Logic of Reactive and Concurrent Systems*. Berlin: Springer.
- Manna, Zohar, and Richard Waldinger. 1976. "Is 'Sometime' Sometimes Better Than 'Always'? Intermittent Assertions in Proving Program Correctness." Memo AIM-281, STAN-CS-76-558. Stanford Artificial Intelligence Laboratory.
- . 1978. "Is 'Sometime' Sometimes Better Than 'Always'? Intermittent Assertions in Proving Program Correctness." *Communications of the ACM* 21, no. 2: 159–72. <https://doi.org/https://doi.org/10.1145/359340.359353>.
- McCarthy, John, and Patrick J. Hayes. 1969. "Some Philosophical Problems from the Standpoint of Artificial Intelligence." In *Machine Intelligence 4*, 463–502. Edinburgh: Edinburgh University Press.
- McDaniel, Kris. 2020. "John M. E. McTaggart." In *The Stanford Encyclopedia of Philosophy* (Summer 2020 Edition), edited by Edward N. Zalta. Stanford University. Accessed November 2024. <https://plato.stanford.edu/archives/sum2020/entries/mctaggart>.
- McTaggart, J. Ellis. 1908. "The Unreality of Time." *Mind* 17, no. 68: 457–74.
- Newcombe, Chris, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. 2015. "How Amazon Web Services Uses Formal Methods." *Communications of the ACM* 58, no. 4: 66–73. <https://doi.org/10.1145/2699417>.
- Øhrstrøm, Peter. 1983. "Richard Lavenham on Future Contingents." *Cahiers de l'Institut du Moyen-Âge Grec et Latin* 44: 180–86.
- Øhrstrøm, Peter, and Per Hasle. 2007. *Temporal Logic: From Ancient Ideas to Artificial Intelligence*. Berlin: Springer.
- . 2020. "Future Contingents." In *The Stanford Encyclopedia of Philosophy* (Summer 2020 Edition), edited by Edward N. Zalta. Stanford University. Accessed November 2024. <https://plato.stanford.edu/archives/sum2020/entries/future-contingents>.
- Pnueli, Amir. 1977. "The Temporal Logic of Programs." In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, 46–57. Providence: IEEE Computer Society.
- . 1979. "The Temporal Semantics of Concurrent Programs." In *Semantics of Concurrent Computation: Proceedings of the International Symposium, Evian, France, July 2–4, 1979*, edited by G. Kahn:1–20. Lecture Notes in Computer Science, Volume 70. Berlin: Springer-Verlag.
- Pratt, Vaughan R. 1976. "Semantical Consideration on Floyd-Hoare Logic." In *SFCS '76: Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, 109–21. Washington, DC: IEEE Computer Society.
- . 1979. "Process Logic: Preliminary Report." In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 93–100. New York: Association for Computing Machinery.

- Priestley, Mark. 2011. *A Science of Operations: Machines, Logic and the Invention of Programming*. London: Springer.
- Prior, Arthur N. 1957. *Time and Modality: Being the John Locke Lectures for 1955-6 Delivered in the University of Oxford*. Oxford: Oxford University Press.
- . 1967. *Past, Present and Future*. Oxford: Oxford University Press.
- Rescher, Nicholas. 1966. *The Logic of Commands*. New York: Dover.
- Rescher, Nicholas and Alasdair Urquhart. 1971. *Temporal Logic*. New York: Springer.
- Simons, Peter. 2021. "Jan Łukasiewicz." In *The Stanford Encyclopedia of Philosophy* (Winter 2021 Edition), edited by Edward N. Zalta. Stanford University. Accessed November 2024. <https://plato.stanford.edu/archives/win2021/entries/lukasiewicz>.
- Yu, Yuan, Panagiotis Manolios, and Leslie Lamport. 1999. "Model Checking TLA+ Specifications." In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, 54–66. London: Springer.
- Zuck, Lenore. 2019. "Amir Pnueli—A.M. Turing Award Laureate." Association for Computing Machinery. Accessed November 2024. https://amturing.acm.org/award_winners/pnueli_4725172.cfm.

Authors

Troy Kaighin Astarte is a lecturer in computer science at Swansea University. Troy graduated with a degree in computer science and a PhD in history of computer science from Newcastle University. Their research is in the history of computing and computer science, with previous work on the history of programming language semantics and concurrency. Troy's teaching is in various areas of computing with a focus on shaping a more socially-aware generation of graduates. Troy is the Editor-in-Chief of IEEE Annals of the History of Computing, a member of the IEEE CS History Committee, and ACM History Committee.

Arianna Borrelli is a historian and philosopher of natural philosophy and modern science working at the Käte-Hamburger-Kolleg "Cultures of Research," RWTH Aachen, where her research focuses on reconstructing the variety of cultures of computer-aided research. She has a special interest for the interplay of scientific knowing and the tools mediating it and has worked on medieval mathematical cosmology, early modern meteorology and mechanics, as well as quantum theories from their early days up to the present. She is currently President of the DHST/DLMPST Commission for History and Philosophy of Computing (HaPoC) and her recent publications include: A. Hocquet, F. Wieber, G. Gramelsberger, A. Borrelli et al. 2024. Software in science is ubiquitous yet overlooked. *Nature Computational Science* 4: 465–8; A. Borrelli. 2023. Aristotelianism, Chymistry and Mechanics in Early Seventeenth-century Europe. In: D. Verardi (ed.). *Aristotelianism and Magic in Early Modern Europe*, Bloomsbury, 105–44.

Nathalie Bredella is Chair of Architectural Theory at Leibniz Universität Hannover and currently Anna Boyksen Fellow at TUM, Institute for Advanced Study. Her research focuses on the history and theory of architecture at the nexus of media and technology studies. Her recent publications include: G. Bailey, N. Bredella, C. Steinitz. 2025. Is That a Computer Program? Carl Steinitz on the Early Years of Harvard's Laboratory for Computer Graphics and Spatial Analysis. *Dimensions. Journal of Architectural Knowledge* 4 (7): 77–29; N. Bredella. 2024. Theorie medialer Architekturen. In E. Christoph et al. (eds.). *Handbuch Medientheorien im 21. Jahrhundert*. Springer, 1–18; N. Bredella. 2022. *The Architectural Imagination at the Digital Turn*, Routledge.

Marie-José Durand-Richard is a historian of mathematics. She has long taught at the University of Paris VIII Vincennes Saint-Denis and she is now Associate Researcher of the laboratory SPHERE, UMR 7219 CNRS & Université Paris Cité. Her special fields of interest are English algebraists and logicians of the first half of the 19th century, historiography of algebra and arithmetic, the history of cryptography, the mechanization of calculus and its applications (planimeters, differential analyzer, computers). She is primarily concerned

with the production of mathematical notations and their role in the meaning of mathematics, for both their designers and their users.

Helena Durnová teaches history of mathematics and computing as well as a course on history of science and technology at Masaryk University in Brno, Czechia. She has written on history of computing in Czechoslovakia in the 1950s and 1960s, including early programming practices there. She is interested in the intimate connections at the intersection of computing, mathematics, and language. She is also working on history of mathematics education in Czechoslovakia and together with Petra Antořová, Danny Beckers, Snezana Lawrence, is preparing the book *A History of Mathematics Education in Czechoslovakia. Ideologies and Practices*, to be published in the Springer series History of Mathematics Education.

Edgar Lejeune is a historian of science, currently a postdoctoral researcher at the CEREDl's Chaire d'Excellence en Édition Numérique (Université Rouen Normandie, France) and associate researcher at SPHERE (CNRS, Université Paris Cité, France). His research focuses on the history of digital humanities and the social sciences, with a special interest in the history of history and historiography. He is notably interested in how computer science and digital tools transform textual cultures in the sciences. Since 2016, he has been a pre- and post-doctoral fellow at the Max Planck Institute for the History of Science, at the École Française de Rome, at the Vossius Center in Amsterdam, at the EHESS and at the University of Angers. His recent publications include E. Lejeune. 2024. 'Open' or 'Close' Research Instruments? Conflicting Rationales in the Organization of Early Digital Medieval History in Europe, *Digital Humanities Quarterly* 18 (2); E. Lejeune. 2022. Objectifs et stratégie d'un bulletin de liaison; Le Médiéviste et l'Ordinateur (1979-1989). *Humanités numériques* [online] 6; E. Lejeune. 2023. Textométrie, histoire et ordinateurs dans les années 1970. Le cas du programme Aline. In L. Dumont, O. Julien, S. Lamassé (eds.), *Histoires de mots*. Éditions de la Sorbonne, 179–97.

Jacqueline Léon is a senior researcher emeritus at the Histoire des Théories Linguistiques research unit, Centre National de la Recherche Scientifique, Université Paris Cité. Her researches, in the field of history and epistemology of contemporary language sciences, focuses on the history of automating linguistics- machine translation, computational linguistics, natural language processing- the history of structuralism and of American and British linguistics of the 20th century. She is particularly interested in the transfer of scientific and technical knowledge and its application. She has built up an archive on the history of automatic language processing (HTAL), comprising around 1,000 classified and archived documents. Recent works include: J. Léon. 2015. *Histoire de l'automatisation des sciences du langage*. ENS Editions; J. Léon, S. Loiseau (eds.). 2017. *Quantitative Linguistics in France*. RAM Verlag; J. Léon. 2021. *Automating Linguistics*. Springer.

Elisabetta Mori is a historian of science and technology. She is Juan de la Cierva postdoctoral research fellow (2024-2026), Universitat Pompeu Fabra, Barcelona. The focus of her academic research is the history of computing in Europe, with a special interest in the 1950s and 1960s. She is an associate editor of the *IEEE Annals of the History of Computing*; a Senior Research Fellow at The UK's National Museum of Computing (TNMOC); a member of the IEEE CS History Committee. She has published her research in a variety of venues: peer-reviewed journals and books, in the *Oxford Dictionary of National Biography*; and in newspapers like *The Guardian*. In 2020 she was guest editor of the *IEEE Annals of the History of Computing* Special Issue *Interface Architects: The Evolution of Human-Computer Interaction*. She is also a professional oral history interviewer, with collaborations in several projects in the UK, Italy, France and the US.

Mark Priestley writes on the history and philosophy of computing and is currently Senior Research Fellow at the UK's National Museum of Computing. His interests include the early history of computing in the UK, the development of programming techniques and languages, and the work of John von Neumann. He has published numerous articles and books on the history of computing, including M. Priestley. 2011. *A Science of Operations: Machines, Logic and the Invention of Programming*. Springer; T. Haigh, M. Priestley, C. Rope. 2016. *ENIAC in Action: Making and Remaking the Modern Computer*. MIT Press; 2018. *Routines of Substitution: John von Neumann's Work on Software Development, 1945-1948*. Springer. His most recent publication is M. Priestley. 2024. Working in the Scientific State: John Womersley's Early Career. *IEEE Annals of the History of Computing* 46(3): 86–93.

Arianna Borrelli and Helena Durnová (eds.)

Computing Cultures: Knowledges and Practices (1940–1990)

Highlighting the diverse and fragmentary nature of the so-called “digital turn,” this volume offers a glimpse into the landscape of different computing cultures which emerged side by side between the 1940s and the 1990s, at times sharing some features, yet remaining essentially independent from each other.

Some of these cultures disappeared, some thrive until today, but understanding all through their knowledges and practices, interconnections and broader historical context, is essential to deal critically with the visions and dreams, fears and tensions characterizing digital practices in today’s knowledge societies.



meson press

ISBN 978-3-95796-273-7



www.meson.press