Logic, Programming, and Computer Science: Local Perspectives

Liesbeth De Mol, *CNRS—UMR 8163 Savoirs, Textes, Langage, Université de Lille, France*

This Special Issue offers new perspectives on the question of how and why logic became intertwined with computing and programming. It focuses on the *locality* of programming and computing practices, highlighting their relation to formalist practices and concepts to provide a deeper understanding of the historical relations between these different practices.

The theoretical and technical considerations that tie logic and computing together have been historically contingent on the local and national structures in which they unfold. The pioneering historian of computing, Michael Mahoney, once noted that "the computer in itself embodies one of the central problems of the history of technology, namely the relation of science and technology" [18]. It follows that the social processes that have guided the articulation of the relationships between logic and computing substantiate and detail how this relation between science and technology is negotiated locally in the practices of computing and programming.

This issue is grounded on scholarly works in computer science, the history of computing, and intersections between the two fields. These works do not form a cohesive framework. By consequence, they have often yielded vastly different—if not conflicting—conclusions on the relationships between computing as a science and computing as a technology and on the role that logic plays in shaping those relationships. In response, this issue suggests that a focus on the locality of logic offers a unique opportunity to study how the rather arcane and abstract field of logic was put into action locally. Methodologically, it requires the integration of internal and external readings to offer a more global, if not longue durée perspective on why and how logic was transposed into the discourse.

COMPUTER SCIENCE PERSPECTIVES

A prominent *belief* among computer scientists states that the technology of computing is founded upon theoretical insights. In other words, as physicist and computer pioneer Nicholas Metropolis had noted, "Theory is the captain, and application the soldier" [20, xvi]. This can lead to the assumption that there are logical foundations of computing that can be historically retraced, directly or indirectly, to the foundational debates in early 20th century mathematics to resolve a number of fundamental logical problems. One school of thought at the time, the so-called formalist school anchored in the work of David Hilbert, hoped to resolve those problems by developing a specific kind of "pure" formalism. According to logician Alonzo Church, this required a formalism that "makes it possible to abstract from the meaning of the symbols and to regard the proving of theorems (of formal logic) as a game played with marks on paper according to a certain arbitrary set of rules" [4, p. 842]. It is here that we find one of the roots of what later became known as the foundational models of computer science: combinators, lambda calculus, general and partial recursive functions, Post production systems, and, of course, Turing machines. While these formalisms were of basic relevance to the field of mathematical logic, for many years, the results were rather arcane and known only to the initiated, which included, among others, John von Neumann.[1]

Today, these formalist views have certainly been influential on some developments in computer science. One prominent view, for instance, that emerged in the late 1960s and 1970s, and which is connected quite directly to the ambitions of Hilbert's program, is the idea that programs can be treated as formal objects for which one can then prove formal correctness. Roughly speaking, this comes down to the idea of proving that a program is correct with respect to its specifications, that is, it can be proven that it is doing what it is supposed to be doing.[2] But this does not necessarily imply that computer science and, with it, the computer, is historically grounded in (or can be reduced to) logic. As is well known among

1058-6180 © 2022 IEEE
Digital Object Identifier 10.1109/MAHC.2021.3121578
Date of current version 7 February 2022.

[1]More specifically, these formalisms allowed us to prove the impossibility of solving certain general problems in mathematics and so resulted in the partial failure of Hilbert's program.
[2]See, e.g., [17] for a strong version of this view.

historians, logic is but one dimension of the history of computer science. In order to understand then why, among certain communities of computer scientists, there exists a conviction that logic *does* play such a basic role for their field, the challenge for the historian is to disentangle the *actual use* of logic within the practice and the theory *from the social forces at work* in the shaping of those practices and theories. For instance, it was only in the 1950s and 1960s that a specific community in computing and programming identified their foundations with logical foundations and integrated logical methods into their work [7].[3] One epistemological explanation for this is the increased need for more controlled methods of programming against the background of a looming software crisis but there is also a strong sociohistorical dimension to be taken into account: that of an emerging new field that struggled with its disciplinary identity and that wanted to achieve independence by providing a mathematical foundation. Clearly, such explanations require an integration of internalist with externalist methods.

While today, such logically founded views are perhaps less dominant than they were in the 1960s and 1970s, there is still a common conception, among a specific group of academic computer scientists, that the origins and foundations of computing are to be found in logic and mathematics. For instance, logician Martin Davis has advocated for the conviction that the abstract model of the Turing machine was basic to the construction of the first computers [8].

Historians do not always accept views such as Davis'. This was clear, for instance, in the wake of the Turing centenary in 2012, a celebration of Alan M. Turing, which reinforced the image that computing is based on logical insights, with misinformed claims that Turing invented the modern computer. Several historical pieces in the U.S. flagship magazine for computer science, *Communications of the ACM*, corrected these claims by showing that they misrepresented the more complex historical reality, and that these views are mostly a side effect of a discipline's need for heroes [5], [6], [13].[4]

HISTORIOGRAPHICAL PERSPECTIVES

Despite the history of computing's tremendous development since historians first approached the subject in the 1980s, the field does not yet have a broad intellectual history of computer science. Mahoney, one of the pioneering scholars, had set out to write such a history in the 1990s, which he provisionally titled *The Structure of Computation: Mathematics and Theoretical Computer Science 1950–1970,* but was never able to finish that project [14].[5] Bill Aspray's *John von Neumann and the Origins of Modern Computing* [1], Donald MacKenzie's *Mechanizing Proof* and Mark Priestley's *A science of operations* [23] have made important inroads in the history of computer science, but a work of the kind Mahoney had envisioned is still lacking.

There are two possible reasons for this. First, while Mahoney certainly adhered also to a more "scientific" reading of the computing field, he concluded that the history of computing has its proper home within the history of technology.[6] The inscription of the main community of historians of computing within the *Society for the History of Technology* has been an affirmation of that viewpoint,[7] though it is clear that in recent years attention for the topic within the history of science is increasing.[8]

A second, complementary reason, is the supposed opposition in the historiography of computing between what is classically called internalist and externalist history. On the one hand, internalist history is often, but not necessarily, associated with people from the field writing their own technical histories. Their critics call these works histories of ideas and rebuke them for being too Whiggish.[9] On the other hand, externalist history is often considered a good practice within historiography, but it does not engage deeply with developments of computing and programming *techniques*.[10]

[3]For instance, as was shown by scholars like Mark Priestley and Matti Tedre. As they both emphasize in their work, this only concerned a particular academic community and should not be generalized to the computing field at large [23], [27].
[4]Also, within the computing field itself, the view that computer science is anchored in logic, is not shared by everyone, and has given rise to several heated debates, if not divides, between those who insist on the need for logical foundations versus those who seek other foundations [11] and/or those who have warned that there are basic pitfalls if one assumes that programs can be treated as mathematico-formal objects [22].

[5]As Haigh explains: "[t]his was the first, and so far the only, attempt to write a reasonably broad intellectual history of computer science" [14].
[6]"What is truly revolutionary about the computer will become clear only when computing acquires a proper history [...] Pursued within the larger enterprise of the history of technology, the history of computing will acquire the context of place and time that gives history meaning" [18, pp. 122–123].
[7]For a historiographic analysis of the relation between the historiography of computing (in relation to the historiography of mathematics) and the particular role of Mahoney's work in this context [10].
[8]This is due to, among others, the increased relevance of the so-called computational science.
[9]Even though it does not necessarily follow that internalist history is Whiggish.
[10]In this context, it is interesting to point out that the Springer book series *History of Computing,* edited by a number of well-known historians of computing, aims at "high-quality books, which address the history of computing, with an emphasis on the 'externalist' view of this history."

This tension between externalist and internalist histories, where the former is associated with "professional" history and the latter with history written by practitioners, became explicit when Donald Knuth, prominent computer scientist, gave a personal reply to a paper by Martin Campbell-Kelly that was published in *Annals* [3]. Knuth's Kailath lecture at Stanford University, titled *Let's not Dumb Down the History of Computing* (2014), regretted that Campbell-Kelly applauded a development within history of computing from being a history driven by technical details to a more external way of writing history. The analysis of this debate by Tom Haigh pointed out quite clearly that the kind of history computer scientists would like to see—that is, one that is engaged with technical details—is not the kind of history-writing valued at U.S. history departments [15]. But it also pointed at a deeper tension between internalist and externalist histories.

TOWARD LOCAL HISTORIES

Within the recent historiography of computing, it has become clear that global histories alone do not account for the richness and diversity of local practices and tend to generalize too much one local perspective to all. This results in too simplistic and one-sided accounts of the history of computing.[11] Within that framework, a growing number of scholars have engaged also more deeply with the history of computer science, by zooming in on local interactions between logic, mathematics, and computing against the background of longer term problems, concepts, and ideals. They engage with the specificities of these contexts—its local technical, social, and scientific challenges—and overcome the too strict distinction between science and technology, on the one hand, and, on the other, that between external and internal analyses. Scholars like Troy Astarte, Maarten Bullynck, Edgar Daylight, Liesbeth De Mol, Stephanie Dick, Thomas Haigh, Elisabetta Mori, David Nofre, and Mark Priestley[12] have traced down *to what extend* certain logical ideals and techniques became transposed *or not* into a set of particular engineering and programming contexts and views [2, 7, 9, 12, 16, 24]. Ksenia Tatarchenko and Pierre Mounier-Kuhn have successfully contextualized and enriched global developments in the history of computer science through an analysis of more local contexts in the case of Russia and France [21], [26].

While such local histories enable us to work against an all too linear, monolithic, and progressive history, it also has a basic pitfall: it does not give a clear answer to how such local perspectives (re)connect to and can be (re)integrated into more global perspectives. Indeed, as Renn pointed out in 1996:

> The emphasis of recent historical studies on the local circumstances of the practice of science have certainly helped us to question the universals superimposed on the history of science by a dogmatic and normative philosophy of science but they should not induce us to consider the microscope alone to be the legitimate instrument of historical analysis, when there are obviously structures that can only be identified with a telescope. [. . .] we should neither think in terms of a strict distinction between internal and external determinants nor in terms of an alternative between local and universal structures governing the historical development of scientific thinking, but in terms of a manifold of structures living on different timescales and crossing—each in its own way—the borders between science and its contexts [25].

Renn found solace in the reliance on long-range studies, which reveal *the "interplay between its local and its more global structures."* Such an approach—telling local histories without assuming that they cannot transcend their own local boundaries—is the key methodological contribution of this issue. Each of the contributions opens up a particular longue durée perspective, by focusing on a specific local context.

LOCAL–GLOBAL HISTORIES

As explained in the "Computer Science Perspectives" section, within computer science, there exists a dominant view that logic provides a foundation for the field of computing not just theoretically but also historically. Such views are often focused on a few pioneering figures who were working in mathematics and logic and then started work on computing machines. In the article "German encounters of logic and programming (1948–1958): Three readings of Turing machines," Maarten Bullynck contextualizes different readings of Turing's 1936 paper [Turing 1936–1937] in a German context (1948–1958). He shows how much a particular reading of the Turing machine is dependent on and determined by local contexts and how the problems and goals that the different practitioners were facing shaped their readings. These can be more

[11]That point was made already by Mike Mahoney with his notion of the so-called "communities of computing" approach for doing history of computing and which proposed to look at different practices from different communities *concurrently* [19].
[12]See, e.g., [2], [7], [9], [12], [16], and [24].

institutional and mathematical (as was the case with Hans Hermes) or more practical and machine oriented (as was the case with Corrado Böhm). Understanding these local contexts requires a more externalist reading of the different cases, whereas the analysis of how the Turing machine was integrated in those works is achieved through a more technical analysis of the Turing machine concept and the different ways it was used to solve or frame a certain problem. Through these different readings, Bullynck shows that one should be careful when inscribing local histories in a more global history, in this case ALGOL, since it risks to create too simplistic histories that do not do justice to the local practices underpinning those very same histories. In other words, by zooming in on one very specific problem "How was Turing read in the German and Swiss context?," the local contexts that had faded in the international history of ALGOL have become visible again. Their chronology tells us of the rapidly moving objectives that drove computing and programming in those days.

In *The Work of Writing Programs: Logic and Inscriptive Practice in the History of Computing,* a different perspective is offered on locality: through the analysis of a number of quite distinct local practices, David Dunning develops a particular way of looking at how logic got entangled with computing. It inscribes programming and logic practices within a broader history of (re) writing, introducing the notion of inscriptive systems. That notion is used as an umbrella term for any kind of notational practice, allowing for an open-ended range of uses for written symbols. It gives a more *general* methodological framework from which to study both logical and programming practices. While each clearly has its own material, social, and technological conditions, that perspective allows us to reveal a connection between logic and computing that renders understandable their historical linkage. The motivation for this lies exactly in trying to resolve the rather strict separation that is often made between mathematics and technology in, among others, Mahoney's work. In the article, mathematical logic, rather than being considered as the immaterial spirit of the computer's material body, is rematerialized as an activity, which always takes place in a physical medium too, thus reconnecting the mathematics and the technology of computing. In order to add strength to that view, the first case of the article is a particular reading of some work on 19th century mathematical logic for which Dunning shows that already at that time, formalist practices were connected to problems of mechanization, thus providing a broader horizon from which to read their later linkage within the context of computing.

In the final article of this issue *Russian Logics and the Culture of Impossible,* which consists of two parts, *Recovering Intelligentsia Logics* and *Reinterpreting Algorithmic Rationality*, the locality of the Russian imperial and Soviet context, relying on a *longue durée* perspective, offers an alternative understanding of the central notion of "algorithm" as it appeared in the Soviet context, countering a dominant and U.S.-centered narrative that introduces "algorithm" as going hand-in-hand with a shift from humanistic ideals toward the so-called cold war (algorithmic) rationality. The two parts of this article show, in contrast, that the notion of algorithm in the Russian context is much more tied to a set of humanist ideals, inscribed in a general vision of educating and shaping minds, and which has its roots in prerevolutionary intelligentia science and logic. The "culture of impossible," as it was named by Vladimir Uspensky, captures that idea: the thinking around and tinkering with mathematical objects to develop, extend, and explore one's own human limitations and capacities. As such, this two-part article shows that also "general" histories are locally determined and require to be complemented and engaged with other "local" histories in order to write a truly general history of, for instance, the algorithm. A central notion is *poznaniye*, which ties together the diversity of *intelligentia logics* in the Russian empire and enables us to embed and explain "algorithm" from this more humanist perspective. *Poznaniye,* in contrast to *znaniye* (knowledge), focuses on the *process* of knowledge acquisition, which is always incompletable. It is here that one finds the roots of a *performative* pedagogical culture and a deep connection between the abstract and the concrete. The fact that "algorithm" as a global notion can at least be partially retraced to the Russian imperial and Soviet context thus opens up alternative readings of the science technology dichotomy within the broader history of programming and computing: from this perspective, there need not be such a strong tension—the changing disciplinary dynamics between logic and cybernetics as developed in part II makes this clear. Moreover, the continuity between prerevolutionary logics and cybernetics shows that influences of logic on computing should not be reduced to developments in Western mathematical logic, and so asks for a broader reading of the history of logic too. Finally, in order to offer this alternative reading, it was necessary to integrate a set of different methodologies and competencies that required both the close reading of a number of logico-mathematical papers to decode the basic continuities with the prerevolutionary *intelligentia logic* as well as a deep understanding of the social forces that enables the kind of continuities uncovered in this article.

ACKNOWLEDGMENTS

This Special Issue is anchored in a roundtable "The introduction of mathematical logic in computing:

Comparing local/national processes" that was moderated by Pierre Mounier-Kuhn at the first autumn workshop of the ANR PROGRAMme project at the University Residential Centre, Bertinoro, Italy. The author would like to thank Pierre Mounier-Kuhn for his initial contributions to this project. The author would also like to thank all the members of PROGRAMme who participated in the roundtable. The work on this issue was partially supported by the Project ANR-17-CE38-0003-01. Special thanks are due to Con Diaz. Not only did he guide me and the authors through the process of getting to final versions by providing very clear comments, suggestions, and advices, but he also learned me a lot about what it means to be a good, open-minded, and engaged editor.

BIBLIOGRAPHY

[1] W. Aspray, *John Von Neumann and the Origins of Modern Computing*. Cambridge, MA, USA: MIT Press, 1990.

[2] T. Astarte, "Formalising meaning: A history of programming language semantics," Ph.D. dissertation, School Comput., Newcastle Univ., Newcastle upon Tyne, U.K., 2019.

[3] M. Campbell-Kelly, "The history of the history of software," *IEEE Ann. Hist. Comput.*, vol. 29, no. 4, pp. 40–51, Oct.–Dec. 2007.

[4] A. Church, "A set of postulates for the foundation of logic (second paper)," *Ann. Math.*, vol. 34, no. 4, pp. 839–864, 1933.

[5] E. Daylight, "A Turing tale," *Commun. ACM*, vol. 57, no. 10, pp. 36–38, 2014.

[6] E. Daylight, M. Bullynck, and L. De Mol, "Why did computer science make a hero out of Turing," *Commun. ACM*, vol. 58, no. 3, pp. 37–39, 2015.

[7] E. Daylight, "Towards a historical notion of 'Turing—The father of computer science,'" *Hist. Philosophy Log.*, vol. 36, no. 3, pp. 205–228, 2015.

[8] M. Davis, *The Universal Computer. The Road From Leibniz to Turing*, New York, NY, USA: Norton, 2000.

[9] L. De Mol, M. Bullynck, and M. Carlé, "Haskell before Haskell: An alternative lesson in practical logics of the ENIAC," *J. Log. Comput.*, vol. 25, no. 4, pp. 1011–1046, 2015.

[10] L. De Mol and M. Bullynck, "Making the history of computing. The history of computing in the history of technology and the history of mathematics," *Revue de Synthèse*, vol. 139, no. 3/4, pp. 361–380, 2018.

[11] P. J. Denning and C. H. Martell, *Great Principles of Computing*. Cambridge, MA, USA: MIT Press, 2015.

[12] S. Dick, "On models and machines," *ISIS*, vol. 106, no. 3, pp. 623–634, 2016.

[13] T. Haigh, "Actually, Turing did not invent the computer," *Commun. ACM*, vol. 57, no. 1, pp. 36–41, 2014.

[14] T. Haigh, Ed. "Unexpected connections, powerful precedents, and big questions: The work of Sean Michael Mahoney on the history of computing," in *The Histories of Computing*, Cambridge, MA, USA: Harvard Univ. Press, 2011, pp. 1–18.

[15] T. Haigh, "The tears of Donald Knuth," *Commun. ACM*, vol. 58, no. 1, pp. 40–44, 2015.

[16] T. Haigh, M. Priestley, and C. Rope, *ENIAC in Action. Making and Remaking the Modern Computer*. Cambridge, MA, USA: MIT Press, 2016.

[17] T. Hoare, "An axiomatic basis for computer programming," *Commun. ACM*, vol. 12, no. 10, pp. 576–583, 1969.

[18] M. Mahoney, "The history of computing in the history of technology," *Ann. Hist. Comput.*, vol. 10, no. 2, pp. 113–125, 1988.

[19] M. Mahoney, "The histories(s) of computing," *Interdiscipl. Sci. Rev.*, vol. 30, no. 2, pp. 119–135, 2005.

[20] N. Metropolis and G.-C. Rota, "Preface," in *A History of Computing in the Twentieth Century*, N. Metropolis, J. Howlett, and G.-C. Rota, Eds., New York, NY, USA: Academic Press, 1980.

[21] P. Mounier-Kuhn, "L'informatique en France de la seconde guerre mondiale au plan calcul," in *L'émergence d'une science*. Paris, France: Presse de l'université Paris-Sorbonne, 2010.

[22] D. L. Parnas, "Software aspects of strategic defense systems," *Commun. ACM*, vol. 28, no. 12, pp. 1326–1335, 1985.

[23] M. Priestley, *A Science of Operations. Machines, Logic and the Invention of Programming*. Berlin, Germany: Springer, 2011.

[24] G. Primiero, E. Mori, and R. Arif, "Validity and correctness before the OS: The case of LEO I and LEO II," in *Reflections on Programming Systems: Historical and Philosophical Aspects*, G. Primiero and L. De Mol, Eds., Berlin, Germany: Springer, 2018, pp. 15–47.

[25] J. Renn, *Historical Epistemology and the Advancement of Science*. Berlin, Germany: Max Planck Inst. Hist. Sci., 1996. [Online]. Available: http://www.mpiwg-berlin.mpg.de/Preprints/P36.PDF

[26] K. Tatarchenko, "A house with the window to the West: The Akademgorodok Computer Center (1958-1993)," Ph.D. thesis, Princeton Univ. Press, Princeton, NJ, USA, 2013.

[27] M. Tedre, *The Science of Computing: Shaping a Discipline*. Boca Raton, FL, USA: CRC Press, 2015.