Computer Science as a Humanities Discipline: *Recovering the Humanistic Roots of Computation*

Michael Piotrowski
Department of Language and Information Sciences
University of Lausanne, Switzerland
michael.piotrowski@unil.ch

Abstract: *Considering computer science as a humanities discipline reframes its purpose: it invites us to understand algorithms and data structures as cultural and epistemic artifacts—expressions of how we conceptualize, represent, and reason about the world.*

1 Introduction

The relation between computer science and the humanities is often framed as one of fundamental epistemological incompatibility. This includes the field of digital humanities, where they supposedly intersect: for Burdick et al. (2012, 16), computation rests on "principles that are […] at odds with humanistic methods." This narrative of a deep divide between computing (quantitative, formal, instrumental) and the humanities (qualitative, interpretive, critical) is reflected in institutional structures and continues to shape academic and public discourse.

Paradoxically, the digitalization of society has not bridged but deepened the divide between what C.P. Snow famously called the "two cultures" (Snow 1964). Contemporary computer science is an increasingly technical, if not antihumanistic, enterprise, while the humanities are expected to analyze such developments only from the outside. This raises a fundamental question: how can we design a curriculum that goes beyond the simple application of computer science methods or their study from the outside, and instead develops computation itself as a humanistic mode of inquiry?

Although seemingly apparent, we believe the dichotomy is false. It obscures a long and rich intellectual genealogy in which computer science arises from humanistic thought, rather than opposing it. This contribution proposes a view of computer science as a *humanities discipline* in its own right, continuing a centuries-old project aimed at formalizing meaning, structure, and interpretation.

2 The Humanistic Genealogy of Computation

The roots of computation lie at the intersection of mathematical logic, linguistics, and philosophy. Its conceptual ancestors—Leibniz, Boole, Frege, Peirce, Turing—were as much philosophers and semioticians as they were mathematicians. From the very beginning, computation was conceived not *only* as the manipulation of numbers, but as the manipulation of *signs*, going back all the way to Llull's *Ars magna*.

In the Continental context, this semiotic orientation found expression in the mid-twentieth century in the form of structuralism. Building on 19th-century precursors, it saw in structure a universal principle of intelligibility. Mainly associated with figures such as Claude Lévi-Strauss and Roland Barthes, their prominence overshadows equally relevant thinkers—Gilles-Gaston Granger, Jean-Blaise Grize, Leo Apostel, Abraham Moles—whose work addressed formal thought in the humanities more directly. As Granger ([1960] 1967, 19) emphasized, formal thinking in the human sciences should be understood "non pas seulement comme réduction des phénomènes aux calculs, mais aussi comme invention de structures nouvelles, voire même d'une mathématique originale." This notion of a *mathématique originale* anticipated what we might today call *computational humanities*: a discipline concerned not merely with automating processes but with inventing formal structures that make meaning analyzable, manipulable, and shareable.

Jean-Claude Gardin's pioneering research in archaeological computing exemplifies this ambition. His attempt to formalize reasoning in the human sciences—from the 1960s to the early 2000s—treated symbolic representation as a means of *rendering interpretive processes explicit*. For Gardin, computation was not a way to mechanize thought, but a *mode of reasoning* that could articulate, clarify, and critique humanistic inquiry.

This genealogy also relates to philosophical traditions that study meaning beyond formal systems, such as phenomenology and hermeneutics. While these traditions rarely figure in mainstream computing curricula, they illuminate how computational structures relate to lived experience, cultural practices, and interpretive

contexts. Their absence from the dominant narrative of computer science contributes to the modern "two-cultures" divide; their reintegration helps reveal computation's longstanding place within the humanities.

3 From Measurement to Structure: The Structural Hypothesis

The structuralist tradition provides a conceptual bridge between computer science and the humanities. Abraham Moles's *hypothèse structurale* (Moles [1990] 1995, 141) formulates the idea that it always possible, and often useful, to consider reality as a composition of identifiable elements, combined according to a finite set of rules, which, together, constitute what is called *structure*. This "structural hypothesis" should not be mistaken for a reductionist program; rather, it affirms that structure is a necessary condition for intelligibility.

It is precisely this insight that also underlies computational modeling, and the computer, as a "modeling machine" (McCarty 2014, 256), allows us to construct, transform, and explore these formal structures. From this perspective, computer science and the humanities indeed share a common epistemological concern: the modeling of meaningful structures, though they have developed under different institutional banners.

To regard computer science as a humanities discipline, then, is to acknowledge that its core practices—modeling, abstraction, formalization—are not alien to humanistic inquiry but are its contemporary forms. As Hamming (1962, vii) observed, "The purpose of computing is insight, not numbers." Computation, at its deepest level, is an interpretive act. Acknowledging the limits of formalization is equally important: gesture, ambiguity, tacit knowledge, and situated interpretation constitute aspects of human phenomena that resist full formal capture yet nevertheless guide the design and use of computational systems.

4 The Undone Epistemology of Computer Science

The epistemological dimension of computer science remains an *undone science*: a field of inquiry that is widely acknowledged as necessary yet insufficiently developed. Mainstream computer science research, shaped by engineering imperatives and commercial incentives, often abstracts away from the interpretive, symbolic, and cultural dimensions of computation. Likewise, much work in digital humanities treats computation as an imported technical instrument rather than as a humanistic mode of thought, thus perpetuating the division between the "two-cultures" divide that, at first glance, seemed to have been overcome.

The consequences of this omission are both theoretical and pedagogical. Theoretically, it leaves unexamined the *conditions of meaning* under which computational models operate: the interpretive assumptions embedded in data structures, the cultural norms encoded in interfaces, the tacit choices that shape modeling languages. These questions belong as much to the humanities as to software engineering. Pedagogically, it leads to computer science curricula that emphasize technical proficiency while neglecting reflexive understanding, and to humanities curricula that rely on computational tools without understanding that they are human artifacts, too.

Understanding computer science as a human science requires addressing this undone epistemology directly: it means examining the formal and interpretive structures that underpin computation, situating them within the broader history of humanistic inquiry, and developing models that account for the complexity, ambiguity, and contextual nature of human phenomena. As Granger ([1960] 1967) reminds us, the challenge is not to reduce human phenomena to calculations, but to *invent new structures and formalisms* capable of doing justice to their richness. This challenge remains largely undone—not because it is impossible, but because it requires traversing disciplinary boundaries and overcoming entrenched institutional expectations about what computer science is "supposed" to be.

5 Towards Computer Science as a Humanities Discipline

Considering computer science as a humanities discipline does not diminish its technical rigor; rather, it reframes its purpose. It invites us to understand algorithms, data structures, and formal systems as cultural and epistemic artifacts—expressions of how we conceptualize, represent, and reason about the world. From this perspective, programming languages function as hermeneutic media, models as theoretical arguments, and computation itself as a form of interpretation.

Such a perspective also reshapes the relationship between theoretical and applied research. As Piotrowski (2024) has argued, applied computational work in the humanities constructs formal models of human phenomena, while theoretical work examines the properties of such models at a higher level of abstraction. The latter, *theoretical computational humanities*, could serve as a conceptual core for a humanistically grounded computer science: a site where formalization and interpretation are developed together rather than opposed.

At the same time, conceiving computer science as a humanities discipline requires attending not only to structure and epistemology, but also to domains such as phenomenology, ethics, and the philosophy of technology. Formal systems do not stand apart from the world; they shape and are shaped by human practices and experiences. A humanities-oriented computer science therefore asks how computational systems frame attention, distribute agency, articulate values, and mediate meaning. It examines not just what computations *are*, but what they *do* in human contexts.

This reframing also clarifies what distinguishes computer science from mathematics and the natural sciences. Whereas mathematics seeks universal structures and the natural sciences aim to describe physical phenomena, computer science *operationalizes* structures as executable processes. Following SICP's notion of *procedural epistemology* (Abelson et al. 1996, xviii), computation provides a framework for expressing knowledge in terms of "how to" rather than "what is": to understand a phenomenon is to formalize the procedure that brings it about. This resonates with Friedrich Engels's idea that we have truly understood a natural process when we can reproduce it. computer science investigates not only the structures we posit, but the behaviors we can make real. This capacity to enact models—turning formal structures into dynamic, interactive, and socially consequential systems, would give computer science a unique place among the humanities disciplines.

Hence a humanistic computer science does more than bringing "values" into computation: it recovers an intellectual lineage in which computation has always been tied to meaning, interpretation, and critique. Reconnecting with this lineage opens a path beyond the "two-cultures" divide.

6 Conclusion

If *undone science* designates research that is left incomplete but worth pursuing, then the humanistic re-examination of computer science is such an unfinished project. The historical and philosophical dimensions of computation—notably developed by structuralist thinkers such as Granger, Moles, and Gardin—remain underexplored in contemporary discourse, overshadowed by technical agendas that privilege efficiency, prediction, and automation.

Recovering these humanistic roots is about re-establishing the historical context, but more importantly about establishing a program for the future. It allows us to understand computation not merely as a technical enterprise but as a *human* enterprise: a way of articulating and transforming structures of meaning. It offers a way to resist the antihumanistic tendencies of present-day computer science and to reconnect the field with broader traditions in the humanities—semiotics, phenomenology, hermeneutics, ethics—that have long examined how structures are shaped by and shape human life.

This reconceptualization also has practical implications. It suggests curricular models that integrate formal reasoning with hermeneutics; encourages research programs that consider formal modeling an integral part of humanistic inquiry; and ultimately calls for the development of institutional computational infrastructures independent of industrial imperatives. Returning to the question raised in the introduction —how to conceive a program that goes beyond applying computer science to the humanities or studying computer science from the outside—this perspective points toward an academically coherent alternative: a computer science grounded in humanistic inquiry, committed to both formal rigor and interpretive depth.

References

Abelson, Harold, Gerald Jay Sussman, and Julie Sussman. 1996. *Structure and Interpretation of Computer Programs*. 2nd ed. MIT Press.

Burdick, Anne, Johanna Drucker, Peter Lunenfeld, Todd Presner, and Jeffrey Schnapp. 2012. *Digital_Humanities*. MIT Press.

Granger, Gilles-Gaston. (1960) 1967. *Pensée formelle et sciences de l'homme*. Nouvelle éd. augmentée d'une préface. Aubier-Montaigne.

Hamming, Richard W. 1962. *Numerical Methods for Scientists and Engineers*. McGraw-Hill.

McCarty, Willard. 2014. *Humanities Computing*. Paperback. Palgrave Macmillan.

Moles, Abraham A. (1990) 1995. *Les sciences de l'imprécis*. Seuil.

Piotrowski, Michael. 2024. "Schism or Renaissance? On the Relationship Between Computational Humanities and Digital Humanities." In *Compendium Computational Theology: Introducing Digital Humanities to Theology*, edited by Christopher A. Nunn and Frederike van Oorschot, vol. 1. heiBOOKS. https://doi.org/10.11588/HEIBOOKS.1521.C21938.

Snow, C. P. 1964. "The Rede Lecture, 1959." In *The Two Cultures: And a Second Look*. Cambridge University Press. https://doi.org/10.2307/1578601.