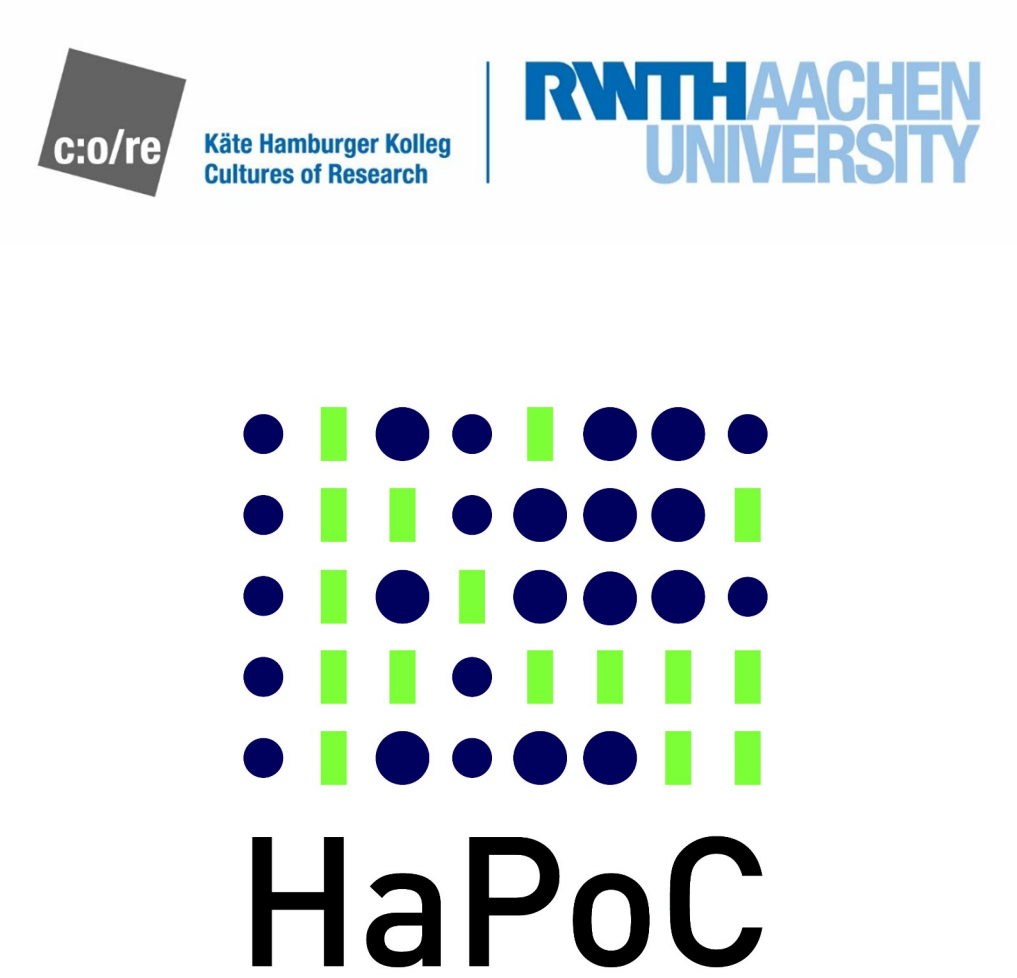

HaPoC 2025

8th International Conference
on History and Philosophy of Computing
17-19 December 2025
RWTH Aachen

Program and Abstracts

HaPoC 2025

8th International Conference on History and Philosophy of Computing
17-19 December 2025
Super C - 6th floor - Ford-Saal
Templergraben 57, Aachen, Germany

and online: streaming.khk.rwth-aachen.de

Wednesday December 17	
9:00-9:30	Opening Words (Gabriele Gramelsberger, Arianna Borrelli)
9:30-13:00	*Session One: New Narratives of Computing*
	9:30 Elisabetta Mori "From *The Mighty Micro* to *The Making of the Micro*: Christopher Evans, Human-Computer Interaction and the popularisation of computing in the UK"
	10:00 Magnus Rust "Of chatbots and hotshots. Social scientists at MIT's Project MAC"
	10:30 *Coffee Break*
	10:45 Kate Mancey "Timbres of potential: Start-up sounds and human-computer relationships"
	11:15 Christina Schinzel "Sigmund Freud in artificial neural networks"
	11:45 *Coffee Break*
	12:00 Troy Astarte "Languages, machines, expressions: Towards an understanding of the discourse of semantics"
	12:30 Gábor Képes "We need a Trabant in computing!"
13:00-14:30	*Lunch Break*
14:30-16:45	*Session Two: Computing Practices in the Sciences*
	14:30 Lara Marziali "Supercomputing and simulation: CINECA's role in shaping the Italian scientific landscape in the 1980s"
	15:00 Marcus Carrier "Learning to fold. The history of neural networks in chemistry"
	15:30 *Coffee Break*
	15:45 Paula Muhr "Computation, human judgment, and epistemic critique: reanalysing black hole images"
	16:15 Jérémy Grosman "The biography of an algorithm: The case of ant colony optimization"
16:45-17:00	*Coffee Break*
17:00-18:00	*Keynote:* Alexandre Hocquet "A portrait of the scientist as a user"
18:00-1930	*HaPoC General Assembly*
20:00	*Conference Dinner* Restaurant "Elisenbrunnen"

Thursday December 18	
9:00	*Welcome and Coffee*
9:30-10:30	*Keynote:* Liesbeth De Mol "What is a computer program? Or, how I liberate(d) myself as a computer user"
10:30-10:45	*Coffee Break*
10:45-13:00	*Session Three: Materialities of Computing* 10:45 Johannes Lenhard "Iterating your way to the future. Mainframe computers and a new culture of prediction" 11:15 Krisztián Németh "Digital archaeology: What a short newsreel reveals about a pioneering Eastern European computer" 11:45 *Coffee Break* 12:00 Clément Bonvoisin "Rage against the materiality of the machines. Encountering and adapting to material constraints in the use of different computing instruments at RAND Corporation (1947 – 1951)" 12:30 Ulf Hashagen "Digital materiality, historical authenticity, museal authority: The program-controlled calculating machines V3, V4, Z4, and Z3 of Konrad Zuse and the Deutsches Museum"
13:00-14:30	*Lunch Break*
14:30-18:00	*Session Four: Histories and Cultures of Programming* 14:30 Amelie Mittlmeier "Committees, working groups, and scientific collaboration: The case of ALGOL 68" 15:00 David Nofre "'There's a strange confusion here': The origins of the Backus-Naur-Form and the challenges of writing the history of early computer science" 15:30 *Coffee Break* 15:45 Mathilde Fichen "From AI to architecture: The performative action of a programming language" 16:15 Simone Martini "A case study in the design of programming constructs: Exception handling" 16:45 *Coffee Break* 17:00 Jad Kadan "Simultaneous code: Multiple invention and the early culture of programming automation, 1947–1952" 17:30 Tomas Petricek "Cultures of programming: The development of programming concepts and methodologies"
18:00-20:00	*Art Event* "The Ambiguity of Sorting" Engagement with artworks (with finger food & drinks) Panel with artists & discussion with Sasha Bergstrom-Katz, Ren Loren Britton, Verena Friedrich Organisation and moderation: Ana Maria Guzmán Olmos

Friday December 19	
9:00	*Welcome and Coffee*
9:30-10:30	*Keynote:* Robin Hill "Desktop to discourse: Philosophy born of Wordstar and VisiCalc"
10:30-10:45	*Coffee Break*
10:45-13:00	*Session Five: Understanding Computational Practices* 10:45 Chirine Laghjichi "Physical computation and the condition of locality" 11:15 Ben Gershon & Oron Shagrir "The mathematical objection to artificial (machine) intelligence" 11:45 *Coffee Break* 12:00 Alexander Gschwendtner "From representation to generation: The epistemic logic of latent space" 12:30 Giora Hon "Who will watch the watchmen (*Quis custodiet ipsos custodes*)? Von Neumann and Turing on computational error"
13:00	*Departure*

Program Committee:
Arianna Borrelli, TU Berlin and RWTH Aachen, Germany
Jianqing Chen, Washington University at Saint Louis, US
Jack Copeland, University of Canterbury, Christchurch, NZ
Beatrice Fazi, University of Sussex, UK
Gabriele Gramelsberger, RWTH Aachen, Germany
Thomas Haigh, University of Wisconsin-Milwaukee, US
Andrei Korbut, CAIS Center for Advanced Internet Studies Bochum, Germany
Alfred Nordmann, TU Darmstadt, Germany
Mitsuhiro Okada, Keio University, Tokyo, Japan
Ben Peters, University of Tulsa, US
Mate Szabo, University of Southern California, US

Local Organising Committee at RWTH Aachen University
Gabriele Gramelsberger (philosophy of science)
Stefan Böschen (sociology of science)
Dawid Kasprowicz (philosophy of science)
Phillip Roth (science and technology studies)
Saskia Nagel (ethics of science)
Torsten Voigt (science and technology studies)
Art event: Ana Maria Guzmán Olmos (philosophy and artistic research)

HaPoC25 - Keynote Abstracts

Wednesday, December 17, 17:00-18:00

Alexandre Hocquet
Archives Poincaré, Université de Lorraine, France

A portrait of the scientist as a user

Why are computational chemists so peculiar ? The history of their relationship with software brings an interesting case study to understand how software may shape scientific activity. We present four key moments in this history — from 1962 to 2024 — to illustrate how visions of openness and user agency have evolved. These include a software-sharing initiative, controversies over licensing and user management, and debates about an AI tool in the field.

The multifaceted category of ‘users’ is key to understand discourses about opennness. We identify patterns of evolution in the relationship between software packages and the computational chemistry community. Throughout our narration, the notion of ‘users’ becomes more complex, in line with the commodification of programs into packages and the increasing complexity of the packages themselves.

We point out that the overall resulting pattern of evolution amounts to a kind of dispossession of scientists’ agency in their relationships with their tools. We propose to view the history of computational chemistry as the formation of a particular ‘repertoire’ where software is central and where the issue of the forms of collaboration and interaction among practitioners implies visions of openness of software development, circulation, maintenance and uses, all in friction.

Thursday, December 18, 9:30-10:30

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

What is a computer program? Or, how I liberate(d) myself as a computer user

The aim of this talk is to present the PROGRAMme project, a collective work amongst researchers from diverse disciplinary and ideological backgrounds who met for several years to work on the question "What is a computer program?". In that project we develop a research programme which assumes it is possible /and/ necessary to work together across disciplinary and other boundaries to turn around a number of fundamental problems we are facing in connection to "programs". In that regard, the project is first of all a humanistic work: while programs have been interpreted before as an exemplification of cold, inhuman rationality, it is clear that more humanistic visions are possible. I present the project from a personal perspective and show how my version of PROGRAMme is deeply anchored in a more activist stance aimed at user liberations which, in my case, goes hand-in-hand with so-called academic nomadisms. I conclude with some concrete proposals for the future of the history and philosophy of computing.

Friday, December 19, 9:30-10:30

Robin Hill
University of Wyoming, Laramie, USA

Desktop to discourse: Philosophy born of Wordstar and VisiCalc

While the American Philosophical Association's Committee on Philosophy and Computers started its mission with community tech help and suggestions, the path proceeded through technique, practice, and application to philosophy. To explore interesting stages along the way, this talk identifies and follows features of office software from their adoption, through their manifestations, to the emergence of philosophical questions. Some are known, some are trivial, but we find questions of potential depth concerning the affordances and constraints of naming, values, separation, and structure, all appropriate for the philosophy of computing.

HaPoC25 – Art Event Abstracts

Thursday, December 18, 18:00-20:00

The Ambiguity of Sorting - How do technologies of separation bring us together?

Art event with contribution by: **Sasha Bergstrom-Katz, Ren Loren Britton, Verena Friedrich**
Organisation and moderation: **Ana Maria Guzmán Olmos**

Digital systems are based on acts of transformation from continuity and unity into discreteness and separation. These acts of transformation have been popularized in the multiple ways in which humans engage with AI systems and other forms of digital technologies. Everything we do is transformed, one could say, automatically into data. It is transformed into discrete units that can be computed. Digital technologies have become more than tools for research; people produce knowledge with them, engage emotionally through and with these systems, generating new forms and spaces of intimacy. At the same time, these technologies carry a history of creating disconnections, failing to account for different ways of engaging with the world. They are the result of a certain imaginary of the human and intelligence, one that leaves aside many other existing ways to be in the world. What kind of communities are enabled by technologies? What kind of technologies are successful in bringing communities together? Can we imagine a history of technology that is based on plural ways of understanding intelligence and the human? How can we design technologies that bring people together? How can we imagine technologies that enable communities?
The Ambiguity of Sorting is a pop-up exhibition and a panel discussion that brings together installation, performance, and conversations to discuss how the history of intelligent systems embodied ideas of intelligence, the human, and how we live together. With this event, we invite to think of a different history of computing, one where those that were not thought to be fitting within their categories were creating their own technologies all along.

Sasha Bergstrom-Katz - ***Unboxing Intelligence***
Bergstrom-Katz's project, Unboxing Intelligence, positions the intelligence test kit not merely as a scientific tool, but as a powerful actor in its own right and casts it at the center of a critical investigation into the construction and dissemination of intelligence. Composed of toys, games, puzzles, and instructional booklets, all organized into a handy kit, the test operates as a charismatic yet destructive object, one that has shaped and circulated ideas of measurable intelligence across the twentieth century and beyond. Approaching it through the lenses of artistic research, performance studies and material cultures, the project theorizes the test as a nesting doll of scientific readymades; it is a mass-produced and ideologically loaded object constituted of readymade theories, objects, and performances. This presentation, specifically, will present test as a readymade performance and will include a short new performance piece.

Ren Loren Britton - ***Indexing Tech for Disability Justice: blocks, boards, plates, shafts & volumes***
Categories produce limits. Limits limit our imagination. Imagination is a resource that we need. We find each other by naming ourselves. What do we call it, let's give it a name. When we go to the trans*crip meeting / club / bar / bookshop / chatgroup / the right doctor / hacking meeting / work session / meetup / class / workshop - we find each other. When wenegotiate more space in categories that confine and open up possibility vis-a-vie broken systems - something changes.
This presentation follows the project: *Indexing Tech for Disability Justice: blocks, boards, plates, shafts & volumes* - negotiating the affordances of naming affinity and sites of embodiminded difference towards an index of technological hir- his - her- story. An Index that would serve an intersectional disability justice tech movement. Moving against a techno- racial- capitalist narrative

assuming that people are only users — to be impacted upon and extracted from — this installation works with a series of historical intersectional crip, trans*, BIPoC innovations that shift the terms of what what we think of when we think of technology.

Verena Friedrich - *ERBSENZÄHLER*

The project *ERBSENZÄHLER* (literally: “pea counter”, idiomatically: “bean counter”) explores the increasing quantification and datafication of life through mathematical-technical procedures and systems – from counting and sorting to statistics and computer-aided processes – and the worldview that accompanies them. It consists of a series of industrial-looking sorting machines in which pea seeds are analyzed, classified, and sorted according to various parameters: from simple, quantifiable features such as weight to more complex ones like color or quality. Each station engages with different aspects of the quantification of life and the history of automation.

In my presentation, I use the *ERBSENZÄHLER* project as an aesthetic and conceptual lens to reflect on broader questions and make these more tangible: What shifts occur when fundamental human activities like counting, classifying, and sorting are increasingly delegated to machines and “intelligent” systems? In this context, I am particularly interested in the tension that arises when algorithmic decision-making replace subjective, situated, and ambiguous processes of human judgment.

HaPoC25 - Talk Abstracts

Wednesday, December 17, 9:30-10:00

Elisabetta Mori, Universitat Pompeu Fabra, Barcelona, Spain and The National Museum of Computing, Bletchley Park, UK

From *The Mighty Micro* to *The Making of the Micro*: Christopher Evans, Human-Computer Interaction and the popularisation of computing in the UK

Dr. Christopher Riche Evans (1931-1979) was a British computer scientist, psychologist, journalist and public intellectual whose work played a role in shaping the popular understanding of science, technology and computing in the UK. Best known for his book *The Mighty Micro: The Impact of the Computer Revolution*, published in 1979, Evans emerged as a key figure in the popularisation of computing and science during the 1960s and 1970s in Britain. His career bridged scientific research and public outreach, making him a mediator between the scientific community and the general public at a time when computers and digital technologies were beginning to reshape society in the UK and everywhere else. In addition, his involvement in experimental literature and computer arts, and his collaboration with writer J G Ballard, makes him a polymath whose intellectual reach extended beyond science and technology into the realms of culture and creativity.

This paper highlights and summarises Evans' contributions to the field of computing, and in particular has a three-fold focus: first, his research in the field of Human-Computer Interaction (HCI) at the National Physical Laboratory (NPL) in Teddington, UK; second, his interest and early research in the history of computing, his collaboration with the London Science Museum and the series of interviews he realised with computer pioneers during the 1970s; finally, his outreach activity as a scientific journalist, and in particular his book and TV documentary based on it, *The Mighty Micro.*

After doing postgraduate research in the Department of Physics, University of Reading, Evans joined the NPL in 1963 in the Autonomics Division; he later became head of the Man-Computer Interaction Pattern Recognition Section Division of Computer Science. As early as the beginning of the 1960s, Evans became aware of the necessity to make computer facilities accessible to everybody, including untrained users; an early figure in the study of HCI in the UK, he was interested in the possibilities that computers opened up for medical applications, the potential for cheap hand-held microprocessor systems, and researched aids for people with health conditions and impairments - the most significant being the MAVIS system, which he developed in collaboration with Loughborough University. The system consisted of a device combining a television viewdata display with a microprocessor and alternative input devices. MAVIS was shown at the Royal Society Conversazione, and it is one of several examples of Evans' contributions to both science and society.

Evans' interest in the history of computing was condensed in the publication of his posthumous book *The Making of the Micro,* published in 1981, and the series of interviews he conducted for the London Science Museum computer pioneers, such as Donald Davies, Konrad Zuse, Grace Hopper among many others. The interviews were published as a series of 20 audio cassettes and made available to the general public by Computer Capacity Management Limited Reading & Hugo Informatics.

In his book*The Mighty Micro* and the six-part documentary series he made with ITV, a British television network broadcast, Evans foresaw and discussed the coming microcomputer revolution and its effect on the economy, industry, and lifestyle of the United Kingdom.

This research is based on papers, articles, interviews and other multimedia original documents, showcasing the authors' latter findings and analysis of materials found in public and private archives, including unpublished materials from Evans' private papers.

References

[1] Christopher Evans, *The Mighty Micro: The Impact of the Computer Revolution*, Gollancz Services, 1979.

[2] Christopher Evans, *The Making of the Micro: A History of the Computer,* Gollancz Services, 1981.

Wednesday, December 17, 10:00-10:30
Magnus Rust, University of Basel, Switzerland

Of chatbots and hotshots. Social scientists at MIT's Project MAC

In 1966, Joseph Weizenbaum published the ELIZA program as part of his work on Project MAC at MIT, which is now considered by some the first chatbot in history and was celebrated at the time as a prime example of AI (artificial intelligence). Project MAC, which introduced the new technology of time-sharing to a broad disciplinary audience for the first time in 1963, is just as wellknown in computer history as Weizenbaum's program with his famous script DOCTOR, which was intended to embody a psychiatrist with whom one could chat via keyboard. However, it is almost unknown that the new interaction possibilities of time-sharing at MIT did not only attract technologists. In addition to various humanities scholars, it was sociologists such as Harold Garfinkel, some curious psychologists and a team of Harvard psychiatrists who were interested in time-sharing and the ELIZA chatbot in particular. The latter used the chatbot for their research at the local Massachusetts General Hospital. The various scientists approached the computer with "experimental epistemologies" to borrow a term from Warren McCulloch. At the same time, Claude Shannon described a "wild orgy of experimental programming" that took place as more and more people had access to computer technologies. While some psychologists wanted to find out whether machines could think, psychiatrists used chatbots to research natural languages, others dreamed of "automated therapy," potentially rendering therapists obsolete. And Garfinkel recognized the human-all-too-human aspect of people in their interaction with computers. The computer became another "technology of the self."

However, it was not only social scientists from outside who wanted to make use of the new technical possibilities. Trained psychologists were already part of the founding generation of computer science that emerged in the 1950s and 1960s. On the one hand, one can think of managers and idea generators such as JCR Licklider, who initiated the financing of Project MAC for ARPA. On the other hand, it was psychologist from Harvard (located like MIT in Cambridge, MA) such as George A Miller and Ulric Neisser who took an early interest in computers and became important figures in cognitive science. Among other things, Neisser published a basic paper on pattern recognition together with Oliver Selfridge in 1960. Neisser was also involved in the research of Project MAC.

An analysis of social scientists and psychologists who worked with and through Project MAC in the 1960s shows that the result of the work was not only exploratory or affirmative. Project MAC's computer systems not only allowed new uses of computing power and for the further developments of computer systems, but also generated its own critique.

Since the end of the 1960s, the AI researcher Weizenbaum had been increasingly critical of his colleagues and their ideas. The MIT philosopher Hubert Dreyfus was hired by the RAND Corporation to take a look at AI research at MIT and wrote a controversial paper. And Neisser, initially open-minded towards AI research, also formulated his critique, which has lost none of its validity to this day.

Wednesday, December 17, 10:45-11:15
Kate Mancey, Utrecht University, The Nethelands

Timbres of potential: Start-up sounds and human-computer relationships
This paper explores the role of music and sound in shaping and reflecting human-computer relationships through analysing the start-up sounds of Apple and Microsoft computers and operating systems, from the single beep of the Apple II in 1977 to the "petals" of Windows 11 in 2021. Far from a mere functional signal, the start-up sound is an integral part of how we, as users, have come to interact with and perceive our computers. In listening closely to these sounds, offering both formal and timbral analyses, this paper highlights the tension between the relative fixity of their function and the evolving musical rhetoric of the various iterations over this almost fifty-year period.

Wednesday, December 17, 11:15-11:45
Christina Schinzel, Universität Padeborn, Germany

Sigmund Freud in artificial neural networks

This contribution aims to reconstruct a moment from the history of computing – more precisely: from the history of machine learning –, when the work of the inventor of psychoanalysis, Sigmund Freud (1856–1939), served as inspiration and as mathematical basis for one of the key functionalities of artificial neural networks. Moreover, this contribution asks what this combined genealogy of psychoanalysis and computer science can tell us about notions of the machinic unconscious.

What does Sigmund Freud have to do with machine learning? Nothing, you might think. But there are more parallels than you might first think. From the 1940s/50s onwards, intelligence became computable. In addition to the classic historical narrative about the emergence of artificial intelligence, in which names such as McCulloch, Pitts, von Neumann, Rosenblatt and Minsky appear, there is also another, far less well-known connection. I refer here to the work of mathematician and computer scientist Paul J. Werbos, who in the late 1980s and early 1990s established the logic of the backward flow of information and control in neural networks (current networks follow the same logic) on the basis of Freud's psychic apparatus and the energy flow that prevails in it. Werbos' work is little known, probably because it coincided with an AI winter and probably also slipped somewhat through the scientific recognition, as he incorporated some borrowings from areas such as psychoanalysis, dreams and Confucianism.

Sigmund Freud's 1895 *Project for a Scientific Psychology (Entwurf einer Psychologie)* served as inspiration, in which the psychic apparatus is dealt with fundamentally, also neurologically, and in which Freud pursued the intention of "providing a natural scientific psychology, i.e. to represent psychic processes as quantitatively determined states of demonstrable material parts, and thus to make them vivid and free of contradictions"[1].

Paul J. Werbos published his research on neural networks in the appropriately titled *International Journal of Approximate Reasoning*, among others. In that journal, Werbos writes: "The original inspiration for backpropagation came from Freud's theory that emotional charge is passed backwards from object to object, with a strength proportionate to the usual forwards association between the two objects." [2] Backpropagation is probably the most important component in all neuronal networks. It is a specific part of the control loop, the back-part of the feedback loop. Backpropagation tries to calculate how the path back looks exactly. It is about a partially determined propagation back to the beginning. The functionalities in neural networks are therefore an attempt to translate Freud's theory into mathematics: "Backpropagation [...] was originally developed as part of a conscious effort to translate Freud's theory into working mathematics."[3]

This contribution aims to reconstruct this hidden psychic influence, which is still at work in today's artificial neural networks, by cross-reading and analyzing literature from computer science and the works of Sigmund Freud. Based on a historical reconstruction and the combined genealogy of psychoanalysis and computer science, this contribution also aims to expand on the notion of the unconscious (a key concept of Freud), as well as current discussions about the machinic unconscious of artificial intelligence.

[1] Freud, Sigmund. *Aus den Anfängen der Psychoanalyse: Briefe an Wilhelm Fliess;Abhandlungen und Notizen aus den Jahren 1887 - 1902*. Frankfurt am Main: Fischer Verlag, 1975, p. 305.

[2] Werbos, Paul J. "Neurocontrol and Fuzzy Logic: Connections and Designs." In: *International Journal of Approximate Reasoning* 6, no. 2 (1992): 185-219, p. 205.

[3] Werbos, Paul J. "Neural Networks and the Human Mind: New Mathematics Fits Humanistic Insight." In: *Proceedings of 1992 IEEE International Conference on Systems, Man, and Cybernetics*, 1992: 78-83, p. 80.

Wednesday, December 17, 12:00-12:30
Troy Kaighin Astarte, Swansea University, UK

Languages, machines, expressions: Towards an understanding of the discourse of semantics

Computing, and computer science, is full of abstraction. It has been argued this is the discipline's defining characteristic (Denning 2025), and that the reification of CS hinged around questions of abstraction (Tedre 2014). How is this abstraction to be understood? In philosophy, abstractions are used by for example Primiero (2019) to structure the ontology of computation; in CS education, very many authors (such as Nakar and Armoni 2023) have discussed how we might impart understanding of abstraction; and historians, as in Abbate and Dick (2022), are beginning to pay attention to the value of dissecting the abstractions used throughout computing's history.

In this talk, I wish to explore another route: analysing the discourse of computer science, specifically its use of conceptuality. The term 'illustrative devices' has been used to refer to metaphors, analogies, figurative names, algorithmic scenarios, and any other uses of non-literal language (Astarte 2023). (Philosophy literature tends to compress discussion of all this into 'metaphor', though I feel these aspects are worth distinguishing.) The study of conceptuality, as argued by Blumenfeld (Friedman 2024), is critical to understanding how knowledge practices navigate into areas of novel discovery, and metaphor is the major tool for managing such encounters with the unknown. Ricoeur (1975) contributed to the philosophy of metaphor with his characterisation of the 'living' metaphor, where the ambiguity of a young metaphor gives it life, and its users participation in the formation of the concept it denotes as a consequence. The major contributors to analysing the role of metaphor in human understanding, however, were Lakoff and Johnson (1980) whose work sparked a subfield between cognitive science and philosophy of language. Their theory of embodied cognition states that humans make sense of any concept only through connection with bodily experience, and it is the deployment of metaphor which creates links between basic understandings and ever-more abstract ones.

One important metaphor from the history of computing is that of the programming language; the origins and impacts of the shift from the various systems of routines and autocodes to the linguistic view of programming is explored in detail by Nofre, Priestley, and Alberts (2014); this adjusted the epistemic nature not just of programming but also linguistics. The period of the 1960s–70s saw a rise in the use of metaphors in computing, as part of the 'high modernism' attitude to science, and an increasingly ambivalent relationship with the computing machine itself (Nofre 2023). This was a key period in the establishment of computer science as a discipline in its own right (Tedre 2014; Astarte 2022), and the use of illustrative devices in this period is the subject I wish to research in detail over the course of the next few years.

In this particular talk, I will present a case study based on the history of programming language semantics, collecting and presenting a taxonomy of example illustrative devices. I will sketch some historical reasons for their development, and begin outlining the effects and implications of these. I intend to show how analysing these illustrative devices can add to our historical understanding of abstraction, the ontology and epistemology of computing, and further how use of illustrative devices may have an impact on our educational practices.

References

Abbate, Janet and Stephanie Dick, eds. (2022). *Abstractions and Embodiments: New Histories of Computing and Society*. Johns Hopkins University Press.

Astarte, Troy Kaighin (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*. Ed. by Janet Abbate and Stephanie Dick. Johns Hopkins University Press.

– (2023). Philosophers, Smokers, and Secretaries: Problematising Illustrative Devices in Computer Science. Talk at panel "Locating Abstraction, Abstracting Power", SIGCIS 23.

Denning, Peter J. (Feb. 2025). "Abstractions". *Communications of the ACM* 68.3, pp. 21–23.

Friedman, Michael (2024). "On metaphors of mathematics: Between Blumenberg's nonconceptuality and Grothendieck's waves". *Synthese* 203.5, p. 149.

Lakoff, George and Mark Johnson (1980). *Metaphors we live by*. Chicago: University of Chicago Press.

Nakar, Liat andMichal Armoni (2023). “On Teaching Abstraction in Computer Science: Secondary-School Teachers’ Perceptions vs. Practices”. In: *Proceedings of the 2023 Conference on United Kingdom & Ireland Computing Education Research. UKICER ’23*. Swansea, Wales UK: ACM.

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”. In: *IEEE Annals of the History of Computing* 45.02, pp. 29–42.

Nofre, David, Mark Priestley, and Gerard Alberts (2014). “When Technology Became Language: The Origins of the Linguistic Conception of Computer Programming, 1950–1960”. *Technology and Culture* 55.1, pp. 40–75.

Primiero, Giuseppe (2019). On the Foundations of Computing. Oxford University Press, USA.

Ricoeur, Paul (1975). *La métaphore vive*. Seuil.

Tedre, Matti (2014). *The science of computing: shaping a discipline*. Chapman and Hall/CRC.

Wednesday, December 17, 12:30-13:00

Gábor Képes, Eszterházy Károly Catholic University, Eger, Hungary an John von Neumann Computer Society

We need a Trabant in computing!

The Homebrew Computer Club (HCC) was an early personal computing community founded in the United States, mainly active during the mid-1970s. According to Theodore Roszak, this club of computer builders represented a countercultural movement that fundamentally reshaped the face of information technology—opening the domain of computing, previously reserved for labcoated professionals, to ordinary people. Computing moved from institutional computer centers to living rooms and hobby workshops. Out of this grassroots movement emerged profit-oriented global companies like Apple and Microsoft. By the 1980s, however, the relevance of the HCC declined in Western capitalist countries, as homebrew computing could no longer compete with mass-produced, professional microcomputers. Behind the Iron Curtain, in Eastern European state-socialist societies, mass-market microcomputers spread with a delay. The HCC served as a valuable model of bottom-up initiative in a centrally planned society.

In my presentation, I explore the Hungarian case. In the mid-1970s, Endre Simonyi, a Budapest based engineer, built a home computer modeled after American designs — at his kitchen table. He eventually traveled to the U.S. and connected with the original HCC, which he sought to adapt to Hungarian circumstances. Beginning in the early 1980s, he ran the Hungarian version of the club within an official scientific association: the John von Neumann Computer Society. The Neumann Society's HCC Division supported the "home cloning" of Western technologies (often under embargo) and promoted the distribution of Hungarian-developed computers, such as the Homelab models, designed by two teenage brothers József Lukács and Endre Lukács. Simonyi persistently lobbied Hungarian Television with his mission: to create a "Trabant" of computing—an accessible, people's computer inspired by the cheap, iconic socialist people's car.

This presentation explores how the Hungarian HCC community functioned within the unique constraints and opportunities of a socialist society. It sheds light on the inventive, "virtue out of necessity" solutions that brought computing closer to citizens in countries where affordable global-market computers were largely inaccessible. In the USSR, such conditions fueled the cult of programmable calculators (Tatarchenko, 2018), while in Hungary—and in countries like Yugoslavia —DIY computer kits remained widely used into the mid-1980s. The Hungarian HCC's Eastern European characteristics also include its social composition: among its members we find young engineers and technicians working for socialist companies who build hobby computers for themselves during their working hours.

My research also addresses the Hungarian HCC's contributions to informal learning and selfeducation. These are considered alongside other initiatives by the Neumann Society in the 1980s: a network of local "microclubs," the *Microcomputer Magazine* (aimed in part at students), and experiments in distance education via national television and radio. These microcomputer clubs played a key role in non-formal education—a phenomenon for which Polish examples have previously been explored (Król, 2022). Beyond presenting the activities and figures of the Hungarian HCC and its representation in the media of the time, the presentation also traces its renaissance. In the 2020s, the revived HCC Retro Microcomputer Division launched computer replica projects and game development contests for the Homelab-2 and Homelab-3—offering a new, popular, and creative approach to the history of computing.

References

Król, Karol (2022). "The Role of Microcomputer Clubs in Education of the Polish Youth in the 1980s: A Retrospective Analysis." *Education Sciences* 12, 150.

Somogyvári, Lajos, Szabó, Máté, Képes, Gábor (2023). "How Computers Entered the Classroom in Hungary: A Long Journey from the late 1950s into the 1980s." In: Flury, C., Gleiss, M. (ed.): *How Computers Entered the Classroom, 1960-2000, Historical Perspectives*, De Gruyter, 39-73.

Tatarchenko, Ksenia (2018). "The great Soviet calculator hack: Programmable calculators and a sci-fi story brought Soviet teens into the digital age". *IEEE Spectrum* 55, 42-27.

Endre Simonyi in the Neumann Society IT History Database https://itf.njszt.hu/szemely/simonyi-endre/

Wednesday, December 17, 14:30-15:00

Lara Marziali, Politecnico di Milano, Italy

Supercomputing and simulation: CINECA's role in shaping the Italian scientific landscape in the 1980s

Numerical modeling and computer simulation have become core methods in current scientific research. This is partly due to the development of computing machines, and partly to the emergence of new modes of knowledge production within science – involving longterm shifts from the Copernican theory that introduced the inductive-deductive method to the hypothetical-deductive research approach of 19th and 20th-century science (Gramelsberger, 2011). Supercomputing lies at the heart of the simulation sciences (Edwards, 2013), and investigating supercomputing centers contributes to the history of scientific simulation and places this within a broader techno-political framework.

In this paper I historically situate CINECA, a supercomputing center created in 1969 and located in Bologna (Italy), within the rise of simulation methods in the 1980s. As stated by Turkle, simulation was not uncritically welcomed by scientists, such as physicists, but was viewed as "something that might taint scientific culture with engineering values" (2009, 40). However, the rise of supercomputing centers such as CINECA suggests a different perspective within the scientific community; scientists pursued the increase of computational power as a means of broadening and enhancing their research. Studying CINECA's historical development shows that physicists were its major advocates and that they were well-represented on its Board and Technical Committee. This indicates that a part of the physics community was not only sensible to but also played a proactive role in shaping the evolution and political status of the supercomputing center. As an interuniversity consortium, CINECA had (and still has) strong connections with universities (especially its major partner, the University of Bologna). Therefore, it is an institution at the crossroads of technical developments, scientific research and governmental interests (but also of CINECA's own political interest in becoming Italy's main supercomputing center). The development of vector processing at the start of the 1980s represented a leap forward in computing capacity – as would the transition to MPP (Massive Parallel Processing). This was not, however, only a question of computing capacity: the role of graphics developments and memory storage are also crucial for the growth of simulation methods. These developments were not always evolving straightforwardly but were the product of various demands from research groups on one end, as well as the chase for technological primacy on the other. In 1990, MURST (*Ministero dell'Università e della Ricerca Scientifica e Tecnologica*) released specific Guidelines for the development of scientific computation and the re-organization of computing centers [1], indicating how important computing and simulation had become also the governmental level. CINECA was envisaged as the main node of this ongoing re-organization, shaping CINECA's predominance in Italy today. By examining CINECA's evolution through the "simulation for science" framework, it is possible to elaborate on Peter Damerow and Wolfgang Lefèvre's argument that the material tools of scientific labor define the possibilities through which scientific abstractions become possible (Lefèvre, 1996). At the same time, these tools do not emerge in a void but are profoundly shaped by the various interests of scientific or political institutions and communities. Analyzing how supercomputing in particular became an aesthetic, material, and hegemonic infrastructure (Larkin, 2013), clarifies how scientific knowledge is shaped by as well as reshapes institutional/political relationships within scientific communities.

[1]A series of initiatives financed by MURST and promoted by the *Commissione per il Calcolo Scientifico*. There were four typologies of initiatives, the first one being high-perfomance computing, the simulation of physical reality in science, engineering and business, etc.

References

Gramelsberger, Gabriele. 2011. *From Science to Computational Sciences*. Diaphanes.

Edwards, Paul N. 2013. *A Vast Machine.* MIT Press.

Larkin, Brian. 2013. 'The Politics and Poetics of Infrastructure'. *Annual Review of Anthropology* 42 (1): 327–43.

Lefèvre, Wolfgang. 1996. 'Tools of Science'. In *Abstraction and Representation*, edited by Peter Damerow, 395–404. Springer.

Turkle, Sherry, ed. 2009. *Simulation and Its Discontents*. MIT Press.

Wednesday, December 17, 15:00-15:30

Marcus B. Carrier, Technische Universität Berlin, Germany

Learning to fold. The history of neural networks in chemistry

Computer simulations in chemistry started as early as the 1950s but were used more regularly from the 1970s onwards. To name just a few developments, in 1970, the computer program GAUSSIAN developed by the group of John A. Pople at Carnegie Mellon University was released as the first commercially available program for computational chemistry. In the 1990s, Density Functional Theory (DFT) changed the way quantum mechanical computations were approximated in computational chemistry. Parallel to these developments that were primarily concerned with making the theory of quantum chemistry usable for chemistry, from the late 1980s onwards, artificial neural networks were used to help predict (at least secondary) protein structures—a development that culminated in the Nobel Prize for AlphaFold in 2024.

When the use of neural networks started in chemistry in the late 1980s, the early adopters very much felt the need to make clear the boundaries of this approach. In the earliest identified article concerning protein folding using neural networks, Henrik Bohr and his co-authors, for example, somewhat apologetically explained: "Although this method [neural networks] lacks the physical transparency of the traditional line of attack, its frank pragmatism is surprisingly successful. Its ability to predict a helix structure appears to be superior to any other method" (Bohr et al. 1988, 223). And a year later, L. Howard Holley and Michael Karplus praised the higher accuracy of protein prediction of neural networks while explaining that this higher accuracy was achieved "despite the fact that the method is naive in the sense that it does not rely on any theory of secondary structure formation, nor does it require a sophisticated treatment of the data from which it is derived" (Holley and Karplus 1989, 155f.). And while Jure Zupan and Johann Gasteiger praised the use of neural networks for predicting protein structures with a higher accuracy than earlier methods in their review article of 1991, they most curiously explained that the theoretical benefits of lied in the field of neural networks not chemistry: "[The use of neural networks for predicting protein structures] opened new perspectives in posing new questions *in the field of neural networks*" (Zupan and Gasteiger 1991, 20).

Chemistry, it seems, was content with pragmatic solutions to practical problems and did not care about theoretical rigor or quantum chemistry too much if the result was practically usable. This mirrors the historiography of the pragmatic research culture in chemistry in the 19th century, when structural chemistry became dominant in organic chemistry without commitments to any realist interpretations of atomism (Rocke 2010). Even in other areas of Computational Chemistry closer related to quantum chemistry, it was a more experimental, pragmatic use that made the simulations useful to working chemists (Lenhard 2014).

As outlined here, I will argue in my talk that machine learning did fit the research culture of chemistry especially well. It was the learning by pattern recognition and the avoidance of direct appeals to theory that particularly made the use of machine learning very akin to chemical thinking.

References

Bohr, Henrik, Jakob Bohr, Soren Brunak, Rodney M. J. Cotterill, Benny Lautrup, Leif Norskov, Ole H. Olsen, and Steffen B. Petersen. 1988. "Protein Secondary Structure and Homology by Neural Networks The α-Helices in Rhodopsin." *FEBS Letters* 241 (1): 223–28

Holley, L H, and M Karplus. 1989. "Protein Secondary Structure Prediction with a Neural Network." *Proceedings of the National Academy of Sciences* 86 (1): 152–56.

Lenhard, Johannes. 2014. "Disciplines, Models, and Computers: The Path to Computational Quantum Chemistry." *Studies in History and Philosophy of Science Part A* 48 (December):89–96.

Rocke, Alan J. 2010. *Image and Reality. Kekulé, Kopp, and the Scientific Imagination*. University of Chicago Press.

Zupan, J., and J. Gasteiger. 1991. "Neural Networks: A New Method for Solving Chemical Problems or Just a Passing Phase?" *Analytica Chimica Acta* 248 (1): 1–30

Wednesday, December 17, 15:45-16:15

Paula Muhr, Brand University of Applied Sciences, Hamburg and Technische Universität Berlin, Germany

Computation, human judgment, and epistemic critique: Reanalysing black hole images

In April 2019, the Event Horizon Telescope (EHT) Collaboration released the first empirical images of a black hole, M87*. In 2022, they released an empirical image of Sagittarius A*, the black hole at the centre of our galaxy. These images, algorithmically reconstructed from sparse, noisy data collected by a global radio telescope array in 2017, marked an achievement once deemed impossible.

The reconstruction process, spanning years, relied on the parallel use of multiple computational methods—some established, others developed for this purpose—guided by the team's expert judgment. The EHT team made their data and algorithms publicly accessible. The reproducibility of these computationally reconstructed images has become a topic of epistemic concern, leading to multiple independent reanalyses of EHT data.

In 2022, five independent studies reanalysed the M87* data to test whether they could obtain sufficiently similar images. These studies employed various approaches: some replicated the EHT team's procedures, while others developed alternative algorithmic techniques. Four of the five reanalyses produced images consistent with EHT's, showing a bright ring-like structure surrounding a central dark area—a visual signature of a black hole shadow initially predicted by Einstein's general relativity. One study, Miyoshi et al. (2022), produced divergent results and was subsequently criticised by the EHT team for methodological flaws. Similarly, Miyoshi et al. (2024) independently reanalysed the Sagittarius A* data, again obtaining an image that diverged from the EHT's reconstruction and arguing that the data were insufficient to support a computational reconstruction of a valid image. The EHT team rebutted this claim, identifying a series of methodological errors in the Miyoshi et al. (2024) analysis.

I argue that each reanalysis, whether convergent or divergent, performed a significant epistemic critique that goes beyond merely reproducing or challenging the initial computational image reconstructions. My analysis draws on Sabina Leonelli's (2018) view that reproducibility serves diverse epistemic functions beyond obtaining the same results. I thus suggest that each reanalysis performed epistemic critique by demonstrating how interpretive human decisions shape the algorithmic processes of image reconstruction from sparse EHT data. Such epistemic critiques are crucial for the epistemological consolidation of black hole imaging as an emerging research field, because they generate methodological insights that can inform future imaging efforts.

While others have highlighted the usefulness of failed replications in exposing the complexity and context-dependence of studied phenomena (Derksen & Morawski 2024), I argue that erroneous reanalyses—which have received little attention—can be epistemically productive. Specifically, I contend that the EHT team's identification of methodological errors in Miyoshi et al. (2022, 2024) not only refuted the critiques but also led to an articulation of methodological constraints inherent to black hole imaging, such as the necessity of employing multiple reconstruction methods in parallel, and generating ensembles rather than single images. Most importantly, the rebuttals foreground the importance of situated expert judgment in guiding the computational image reconstruction from sparse EHT data. These cases demonstrate that especially erroneous reanalyses can catalyse discussions that enable the articulation of methodological specificities and constraints, thus strengthening the research community's understanding of the epistemic limits of the novel field of black hole imaging. Reconstructing images from sparse and variable data is not a straightforward technical procedure but one shaped by modelling assumptions, algorithmic regularisation, and expert judgment. Open discussion of these interpretive elements—rather than attempts to eliminate them— is essential for the epistemological consolidation of black hole imaging as a field.

Wednesday, December 17, 16:15-16:45
Jérémy Grosman, Université de Namur, Belgium

The biography of an algorithm: The case of ant colony Optimization

The present paper proposes a philosophical history of an algorithm: 'Ant Colony Optimization' – from the late 1980s to the late 2000s. Myrmecologists observed long ago that ants were quite good at finding short paths from nest to food. They more recently established that, within some species, some ants do so by leaving trails of pheromones behind them, by kicking their abdomen on the ground, whenever they find a good food source. Jean-Louis Deneubourg, a biologist initially trained as a chemist under the Nobel Prize winner Ilya Prigogine at the Université Libre de Bruxelles, was among the first to devise empirically successful mathematical model of ants' foraging behavior. Deneubourg's natural ants appealed engineers in three manners at least. First, ants were described through a set of rather intuitive differential equations (e.g. ants probabilistic choice functions, amount of pheromones on th paths, etc.). Deneubourg's ants interested engineers insofar as they appeared more 'artificial' than most biologists' 'natural' ants. Second, ants were described with a conceptual apparatus that made them an instance of a 'self-organizing' process. Deneubourg's ants interested engineers insofar as they were presented as displaying a form of collective intelligence. Third, ants seemed to be confronted to problems rather similar to those computing engineers had to address (e.g. finding the shortest path in a networks passing through all its nodes). Deneubourg's ants interested engineers insofar as they could help them to address optimization problems. Let us now see how exactly engineers came to be interested in these ants.

Marco Dorigo, then doctoral student in engineering at Politecnico de Milano, always worked at the crossroads of 'robotics' and 'metaheuristics' – I shall here mostly be interested by the later. In a nutshell: engineers in metaheuristics seek to invent algorithms capable of finding good enough on a *large* range of optimization problems – typically: engineers hope to find *one* general way of addressing *various* problems, where exact or approximate algorithms fail. Dorigo took inspiration from Deneubourg's modeling of ants' foraging to develop 'Ant Colony Optimisation'. Like most engineers he indisinctly talk of 'analogies' or 'metaphors'. The matter is however sufficiently clear: Dorigo's ants essentially borrowed to Deneubourg's ants their mechanism for storing information about the quality of solutions (pheromones) upon the problem structure (paths) that seemed to enable them to address efficiently a variety of situations. 'Ant Colony Optimization' initial emergence and successive mutations shall enable me to bring forward three claims about the productive role of the ant's metaphor in engineering practices. First, the metaphor helped engineers technically: enabling them, along the years, to imagine novel technical schemes that, in some cases, proved particularly fruitful – i.e. outperforming established algorithms on standard benchmarks. Second, the metaphor assisted engineers cognitively: leaving traces in their vernacular descriptions or their source codes, ants appeared to provide valuable support for discussing their algorithms. Third, the metaphor provided an important conceptual resource for speculating about 'artificial intelligence' – distinct from the cerebral or evolutionary metaphors that dominated artificial intelligence since the sixties.

The case presented here is essentially based on available publications , archival materials as well as oral history interviews. The case, I believe, presents a broader interest, both, historical and philosophical. From a historical perspective, the account provided brings us to partly alter the received picture of artificial intelligence. First, it documents the little known field of metaheuristics, once deemed central to 'artitifical ingelligence'. Second, it invites us to conceive artificial intelligence as a speculative investigation offering to sound our notion of intelligence by designing machines as well as a practical enterprise for finding general solutions to various problems. From a philosophical perspective, the paper provides valuable insights into the roles metaphors play in computing practices. First, it accounts for the importance of metaphors in computing practices, insofar as they foster technical imagination and provide cognitive assistance. Second, it highlights some of the conditions that must be met for a metaphor to have productive effects in computing practices: contingent articulations concretely bridging the distance separating the two fields as well as the distinction between a mere technical mutation and a genuine invention. Hopefully thoses considerations might as well be of interest to some of the puzzled engineers working in metaheuristics.

Thursday, December 18, 10:45-11:15

Johannes Lenhard, RPTU Kaiserslauten, Germany

Iterating your way to the future. Mainframe computers and a new culture of prediction

There could hardly have been a more dramatic setting for the publication of a scientific study than that of "The Limits to Growth" (LtG, Meadows et al. 1972). Commissioned by the Club of Rome, it was presented to a select audience of scientists and politicians gathered at the Smithsonian Institution. The message was alarming: if economic growth, environmental pollution, and population growth continued at their current rates, the world would collapse in less than a century. The news spread rapidly, informing the general public about the predicament facing humanity.

In a sense, this message was nothing new. Warnings about the end of the world show that predictions about the fate of the world are almost as old as history itself. Nevertheless, "The Limits to Growth" was different because it was a scientific prediction based on a computer model. On the political side, it is often seen as the starting point for an environmental movement that combines political action and scientific predictions.

Based on Johnson and Lenhard (2024), this talk takes LtG as an example of an emerging culture of prediction that is both iterative and numerical in nature. They refer to works such as Akera 2007, Haigh et al. 2016, Heymann et al. 2017, or Edwards 1996, which deal with the history of large digital computers and their rich technological, social, and institutional context.

The talk considers two perspectives:

(i) On the scientific side, the LtG study marks the culmination of system dynamics, pioneered by engineer Jay Forrester (1918–2016), inspired by modern computers, and claiming to be in possession of the right method for dealing with complexity. The study was a product of a new type of mathematical modeling. While maintaining the goal of prediction, it geared mathematics toward iteration and complexity. Forrester's work is a critical piece of history that shows, as if under a magnifying glass, how technology, mathematical tools for making predictions, institutional contexts, and conceptions of the future have coevolved.

(ii) At the same time, LtG marks a turn after which thinking about the future was perceived increasingly as an activity inside the mainframe culture of prediction. The contribution examines how thinking about the future was being channeled toward computer modeling and advancing under the umbrella of a mainframe culture of prediction. From this perspective, the discussion around LtG can be located in the futurism of the 1960s and early 1970s.

By integrating these two perspectives, the lecture raises the following question: Does this establish the future as a legitimate scientific subject – or rather eliminate it?

References

Akera, Atsushi: *Calculating a Natural World. Scientists, Engineers, and Computers During the Rise of U.S. Cold War Research*, The MIT Press, 2007.

Edwards, Paul N.: *The Closed World. Computers and the Politics of Discourse in Cold War America*. The MIT Press, 1996.

Forrester, Jay W.: *World Dynamics*. Wright-Allen Press, 1971.

Haigh, Thomas, Mark Priestley, and Crispin Rope: *ENIAC in Action. Making and Remaking the Modern Computer.* The MIT Press, 2016.

Heymann, Matthias; Gramelsberger, Gabriele, and Mahony, Martin (eds.): *Cultures of Prediction in Atmospheric and Climate Science*. Routledge, 2017.

Johnson, Ann and Johannes Lenhard: *Cultures of Prediction. How Engineering and Science Evolve with Mathematical Tools*. MIT Press, 2024.

Meadows, Donella H., Dennis L. Meadows, Jorgen Randers, William W. Behrens III: *The Limits To Growth. A Report for 'The Club of Rome's' Project on the Predicament of Mankind*. Universe Books, 1972.

Thursday, December 18, 11:15-11:45

Krisztián Németh, Department of Telecommunications and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary

Digital archaeology: What a short newsreel reveals about a pioneering Eastern European computer

In the late 1950s, Hungarian telecommunications engineer and university professor László Kozma (Képes 2014) wanted to teach his students the principles of digital logic, which were used to control telephone exchanges. He concluded that the most effective way to do this was to build a functioning computer. The result was the MESZ I—the very first electromechanical computer in Hungary—constructed between 1957 and 1958 from relay components. It operated for nearly a decade before being transferred to a museum. The MESZ I was a one-of-a-kind machine. It employed approximately 2,000 relays for both the central processing unit and memory. It could store just 12 numbers, each represented by 33 bits. Programs were read instruction-by-instruction from custom-made punched sheets, marking its architecture as non-von Neumann. Numeric inputs could be entered during runtime, and the output was printed by a modified electric typewriter—already about 30 years old at the time. Throughout the world, few relay-based computers were ever built, and even fewer have survived. Although the MESZ I is well preserved, its cabinet interconnections have been desoldered. Very little documentation remains, most notably an English-language scientific paper—remarkable given the context of 1950s Eastern Europe—authored by Kozma himself (Kozma 1959). However, none of it is sufficient to fully reconstruct the machine's operation or instruction set. No program sheets have survived either, and there is hardly any written information on what programs were run on the MESZ I.

Although restoring the machine to full working order seems unrealistic, developing a software emulator and rebuilding parts of the machine would be feasible if sufficient documentation existed. It seemed, however, that no such records had been preserved. After years of unsuccessful searches for additional documents or knowledgeable individuals, our team decided to pursue a more demanding but potentially more fruitful approach: reverse-engineering the entire machine.

The task might sound simple: determine how each relay operates and identify all connections between them. But the challenge is immense—tens of thousands of connections, each to be discovered manually using a simple multimeter. We explored more advanced techniques, but none proved more effective than painstakingly 'beeping out' connections. The project began about a decade ago and has gained significant momentum in recent months. We estimate we are well over halfway, having even constructed a reduced-capacity working model of the memory.

Reverse-engineering and consulting witnesses are not our only sources of insight. A 60-second 1958 newsreel showing the machine in operation was discovered, but its initial resolution was too low to see fine details like printed numbers. Fortunately, we were later able to obtain a high-resolution digital copy from the Film Archive of the National Film Institute, Hungary. The newsreel clip contained 1,445 video frames. While that may not sound like much, we were surprised by how much we were able to deduce and learn from it. Firstly, our frame-by-frame analysis enabled us to reconstruct a program sheet. For decades, there were no remaining sheets for the MESZ I, but we were able to recreate one. We used the same techniques as Hungary's first programmers, acquiring old chest X-ray films, washing away the bone images, cutting the sheets to size, and punching holes with a hammer and leather punch. We also learned from the short movie clip which programs were first run on the machine. These included solvers for 2×2 and 3×3 linear systems, as well as routines for quadratic and cubic equations. Kozma's paper describes the cubic equation solver specifically, though the newsreel reveals that its actual implementation differed slightly.

One especially revealing detail was the visible program counter: a row of small lamps that lit up in sequence as each instruction was executed. A 4.2-second segment (only 101 video frames) shows these indicator lights. By tracking when each lamp lit and how long it stayed on, we inferred the machine's execution speed. Even more significantly, we identified repeated operations and

correlated them with our reconstructed program sheet, allowing us to deduce which bits of the 12-bit instruction word represented the operation code and which the address.

Building this machine in the 1950s Hungary was a pioneering technological achievement. In many ways, our reverse-engineering follows in the same footsteps—we invent methods, create tools, and proceed with few precedents. Our work continues. Having retraced over half the wiring and understood these sections, we are increasingly optimistic about this digital archaeological research.

References

Képes, Gábor (2014). *Kozma László*, In: Krámli, Mihály; Csáki, Krisztina (ed.) *Sorsok és találmányok.* Magyar Műszaki és Közlekedési Múzeum pp. 70-85.

Kozma, L. (1959). *The New Digital Computer of the Polytechnical University Budapest.* Periodica Polytechnica Electrical Engineering, 3(4) pp. 321–343

Thursday, December 18, 12:00-12:30

Clément Bonvoisin, *Laboratoire SPHERE* (Université Paris Cité – Université Paris 1 Panthéon-Sorbonne – CNRS) and Université Paris 8

Rage against the materiality of the machines. Encountering and adapting to material constraints in the use of different computing instruments at RAND Corporation (1947 – 1951)

Recent scholarship in the history and philosophy of computing has questioned the material constraints at play when implementing abstractions on electronic digital computers. In this communication, I wish to address the following, related question: how does the materiality of different computing instruments shape the way computations are prepared, organised, performed and analysed? To do so, I will carry out a compared analysis of how the same computations were performed by the same people on two different computing instruments.

More precisely, I will focus on the works carried out by mathematician Magnus Hestenes (1906 – 1991) and electrical engineer Arnold Mengel (1919 – 1986), from 1947 through 1951. Back then, both were employees of a recently created organisation funded by the U.S. Air Forces, the California-based RAND Corporation. They worked together on a project aimed at providing guidelines to fighter pilots on how to fly aircrafts, by computing so-called optimal trajectories of the aircrafts from the perspective of the pilot. At that time, RAND Corporation's computing department did not yet possess its well-known electronic digital computer, the JOHNNIAC.

It did nonetheless have three types of computing instruments. Indeed, aside from desk calculators, the department had an IBM 604 and a modified version of the first electronic analogue computer commercialised in the United States, the Reeves Electronic Analogue Computer (REAC). These computing instruments operated differently. The IBM 604 was an electromechanical digital computer, relying on data provided on punched cards and instructions wired using a plugboard to perform arithmetic computations. The REAC was an analogue computer: to study a given system, an operator would wire its different basic components on the plugboard of the REAC, in order to assemble an electronic circuit that behaved as the system under study would.

Archival sources from RAND Corporation on both the project and the use of the computing instruments allow me to reconstruct how Hestenes and Mengel tried to tackle the optimal aircraft trajectory problem. In this talk, after an explanation of how they modelled the optimal trajectories, I will focus on how they approached the use of the different computing instruments they had at hand to compute them. More specifically, I will contrast the different techniques Hestenes and Mengel had to develop in order to use the REAC and the IBM 604. On the one hand, I will show how they were compelled to carry out mathematical approximations in the model to adapt to the materiality of the REAC. Using these, and owing to the speed of the electronic analogue computer, they were able to "get a feel for trajectory optimization." The poor accuracy of the REAC however limited any further conclusion on the phenomenon under study. This limitation could theoretically be overcome with the IBM 604, a more precise computing instrument indeed. Yet, the discretisation required by the punched card system of the IBM 604 forced Hestenes and Mengel to devise an *ad hoc* algorithm to seek the optimal trajectories – only to be in turn limited by the poor speed of the IBM 604.

Thursday, December 18, 12:30-13:00

Ulf Hashagen, The Research Institute for the History of Science and Technology, Deutsches Museum, Munich, Germany / Ludwig-Maximillians-Universität Munich, Germany

Digital materiality, historical authenticity, museal authority: The program-controlled calculating machines V3, V4, Z4, and Z3 of Konrad Zuse and the Deutsches Museum

The concept of 'authenticity' has been utilized with increasing frequency in historical studies over recent years, both from a methodological perspective and with regard to the content of research. While the archive constitutes the primary space of the authentic for historians, as the authentic documents being kept there constituted the main sources and the empirical basis for historical research, this is completely different in the public perception, as the museum is regarded as the place or institution of the authentic par excellence. In this context, the lecture poses the question of whether the authenticity of historical artifacts is a relevant consideration for research in the history of computing from a methodological perspective. To elaborate further, the lecture will address the extent to which the issue of the historical authenticity of digital hardware artifacts exerted an influence on the history of computing.

The following discussion will be based on a case study, with a focus on four computer artifacts, namely the program-controlled calculating machines V3 (1940-1942), V4 (1942-1945), Z4 (1949-1950), Z3 (ca. 1960-1962) built or rebuilt and redesigned by the German computer inventor Konrad Zuse (1910-1995) with the help of colleagues or employees of his companies. The two artefacts which have survived to the present day, namely the Z4 and the Z3, are part of the Deutsches Museum's collection in Munich (Germany). Since its foundation the Deutsches Museum has evolved into a distinguished institution in the realm of technology museums, playing a significant role in the preservation and exhibition of artefacts that chronicle the technological advancements of the past.

The lecture will firstly briefly recapitulate the often-told story of the inventor Konrad Zuse and the V3 (“Versuchsmodell 3”) and V4 (“Versuchsmodell 4”) calculating machines he built. It will try to place the “digital materiality” of these two artifacts in the historical context of the Nazi Reich's military-industrial complex. I will argue that Zuse made extensive use of the resources of the military-industrial-scientific complex for the development of his computers and put his his company at the service of the armaments industry of the “Third Reich”. After Zuse's contacts with armament research institutes led to the construction of an experimental model V3 relay computer in 1941 he was commissioned to build a fully operational V4 computer financed by the Reich Aviation Ministry. In 1945 Zuse and the V4 were finally relocated to the Mittelwerk underground armaments center, where he joined the relocation of missile specialists from Peenemünde to the so-called “Alpine Fortress”. I will further argue that the reconstruction and modification of the V4 initiated by the Swiss mathematician Eduard Stiefel in 1949 not only changed the characteristics of the machine, but created a “new” computing machine. The loan of the renamed machine (Z4) in 1950 enabled Zuse to found a new company in Hesse and to create a new identity as company owner in West Germany.

Secondly the talk sets out to examine the provenance of these two artifacts and the role of the Deutsches Museum and Konrad Zuse in their acquisition and subsequent inclusion in the museum's collection. The manner in which the alleged historical authenticity of the artifacts was employed by the museum and the inventor Konrad Zuse for their respective intentions and goals will also be examined. In the early 1960s, Zuse's objective was to demonstrate, with the assistance of a replica of the V3, now designated the Z3, that he was the true inventor of the computer. The authority of the museum was to provide him with the necessary support to achieve this objective. Conversely, the museum's objective was to acquire a replica of the first computer, authorized by the inventor, at least as a replica in its collections and to exhibit it in its exhibitions. Both parties seemed to have agreed that the war-related background of the V3 and V4 machines and their origins in the military-industrial complex of the Third Reich should be pushed back as far as possible and the machine should be given a new identity.

Thursday, December 18, 14:30-15:00
Amelie Mittlmeier, Ludwig-Maximillians-Universität Munich, Germany

Committees, working groups, and scientific collaboration: The case of ALGOL 68

Since at least the mid-20th century, the scientific committee has become a central institutional form for expert collaboration. Whether embedded in international scientific organizations, government advisory bodies, or standardization efforts, committees have served as structured platforms for negotiation, coordination, and decision-making. While such bodies facilitated the transnational circulation of knowledge and helped establish disciplinary norms, they were - and remain - ambivalent in public perception. Often seen as slow or opaque, committees are frequently regarded as necessary yet inefficient collaborations in scientific endeavors.

In the emerging field of computer science, which began to take shape around 1950 and was institutionalized at US and European universities from the 1960s onwards, scientific committees played a prominent role, too. Key programming languages such as ALGOL 60 and COBOL were developed through committee-based collaboration. While the ALGOL committee sought to create an international standard – assigning members as national delegates - the COBOL committee was oriented toward unifying expertise from different application domains.

Despite the technical successes of such collaborations, getting to results was rarely a smooth process. The committees involved had to contend with logistical hurdles, conflicting priorities, and a lack of shared methodological assumptions. Participants frequently found themselves negotiating not only technical specifications but also fundamental questions of authority, legitimacy, and expertise. Moreover, the requirement to reach consensus often clashed with prevailing scientific ideals such as objectivity, optimality, and rigor. These tensions raise important questions about the relationship between institutional forms of collaboration and the epistemic norms of science.

This presentation explores these questions through a case study of the development of ALGOL 68. In 1962, responsibility for further development of the ALGOL language was assigned to the Working Group 2.1 of the newly founded International Federation for Information Processing (IFIP), an umbrella organization that sought to coordinate computing research at a global level. Notably, this group deliberately avoided the term “committee,” opting instead for a “working group”.

Between 1962 and 1968, the members of Working Group 2.1 engaged in a complex and, at times, contentious process of specification, debate, and revision, culminating in the release of the ALGOL 68 report. This talk argues that, despite their terminological shift, the working group encountered challenges similar to those faced by its committee predecessors. Drawing on archival materials, meeting minutes, correspondence, and contemporary publications, this presentation reconstructs the dynamics of collaboration within the group. It asks: How did the participants organize their work? What decision-making procedures did they establish? What rules - formal or informal - governed their interactions? And to what extent did their self-understanding as scientists shape, and become challenged by, the collective nature of the project?

By focusing on the development of ALGOL 68, the presentation contributes to a broader understanding of committee work as a scientific practice. Far from being a peripheral or purely administrative function, committee work emerges here as a critical site where epistemic values, disciplinary boundaries, and scientific authority were actively negotiated - and, in doing so, shaped the evolving research field of computing.

Thursday, December 18, 15:00-15:30
David Nofre, Independent Scholar, the Netherlands

"There's a strange confusion here"[1] The origins of the Backus-Naur-Form and the challenges of writing the history of early computer science

The aim of this talk is to reflect on some of the historiographical challenges that historians of computing face when seeking to historize the nature of computer science. With this aim in mind, I will use as a case study the contested intellectual origins of the Backus-Naur form (BNF), also known as Backus Normal Form. John W. Backus from IBM developed first this notation in 1959 to describe in a precise way the construction rules of valid statements in the IAL programming language, a language for scientific computation today better remembered by its later name Algol 58. Backus' notation has gone several revisions and extensions over the years, becoming a standard tool in the design of new programming notations. But the context and intellectual milieu in which Backus' contribution took shape has taken on something of the guise of a historiographical puzzle, not least due to Backus' own conflicting recollections.

Specifically what is at issue is the possible influence that the work of mathematical logician Emil L. Post in the period 1920-1950 had on Backus. So far this possible influence has been discussed from a history of ideas approach, a discussion in which Noam Chomsky' own debt to Post's ideas also plays a role. I will instead focus on the context of production of Backus' work, and in particular on the serious situation in which IBM, Backus' employer, found itself due to the appearance of IAL. Indeed, so critical was the situation that IBM contemplated the possibility that Algol would replace its own Fortran language. This raised the question at IBM of how to efficiently machine-convert customer Fortran programs into Algol programs on a service basis. Backus' work must be thus seen in this light, as an effort firmly implanted in the economics of computer programming. This suggests that the flight to abstraction and away from the physical machine that characterized the rise of computer science was itself at least partly rooted in the reality of labor efficiency and economic viability.

In my talk I will also point at the historiographical challenges that make it difficult, but not impossible, to reconstruct the complex interplay between computer programming, linguistics, and logic that might have shaped Backus' work. In particular, I will discuss the important role that informal scholarly networks played at that time, many of which left behind little archival record; the loose citation practices that characterize this period; and the rapid pace at which the transformation of computer programming took place. Despite these difficulties, the case of Backus' notation can help us to appreciate the extent to which the appropriation of mathematical logic within computer programming was more selective, instrumental, and driven by intuition, than is commonly assumed.

[1] John W. Backus as quoted in Dennis E. Shasha, Cathy Lazere, *Out of their minds: the lives and discoveries of 15 great computer scientists* (1998, p. 17)

Thursday, December 18, 15:45-16:15
Mathilde Fichen, Conservatoire National des Arts et Métiers - HT2S, Paris, France

From AI to architecture: The performative action of a programming language

The Prolog programming language was developed in the early 1970s by the Artificial Intelligence Group at Marseille-Luminy University and the Artificial Intelligence Department at the University of Edinburgh. The language introduced the logic programming paradigm, which proved particularly well-suited to symbolic artificial intelligence applications. Rather than instructing the computer through sequential commands, programming in Prolog involves describing the user's world in first-order logic, delegating to the computer the task of deriving appropriate procedures in response to user queries.

Following the perspective of MacKenzie and Wajcman (MacKenzie and Wajcman, 1999), we argue that a programming language is not only shaped by society but also contributes to shaping society in return. In this paper, we examine a specific case: the adoption of Prolog by the Computer Aided Architectural Design department (EdCAAD) at the University of Edinburgh, in order to demonstrate the performative power of programming languages.

In 1979, a C interpreter for Prolog was developed for the PDP-11 computer at EdCAAD, a research group within the Architectural Research Unit, founded in 1968 by Aart Bijl to explore the relationship between computing technology and architectural design practices.

As a first step, Peter Swinson, a researcher at EdCAAD, conducted a comparative study of Prolog, FORTRAN, and C. This led in 1980 to the publication of a conference paper titled Prescriptive to Descriptive Programming: A Way Ahead for CAAD (Swinson, 1980). The team subsequently developed a series of experimental applications in Prolog, ranging from geometric modeling tools to a program that generated stair layouts. In 1984, EdCAAD received funding to develop the MOLE system (Modelling Objects with Logic Expressions). The system aimed to give designers more freedom by allowing them to manipulate either plan drawings or textually entered information interchangeably.

This paper draws on the EdCAAD archival collection, preserved at the University of Edinburgh. The broader influence of AI on computational architecture has been studied by Gaudillère-Jami in her PhD dissertation (Gaudillère-Jami, 2022). By focusing on the migration of Prolog from an AI research context to EdCAAD, and its adoption by the latter, we aim to demonstrate the three points below.

First, the adoption of Prolog at EdCAAD was guided by an evolutionary vision of programming languages. The team viewed programming languages as progressing toward higher levels of abstraction, moving away from low-level interactions with computer hardware and closer to the cognitive models and working practices of human users. This trajectory aligned with the aspiration to create tools that could eventually be used by non-technical individuals.

Second, the case of EdCAAD reveals an inversion of means and ends in technology adoption. Prolog was not selected to meet a specific technical requirement or design problem. Instead, it was embraced because of the conceptual worldview embedded in the language itself. The department's applications were subsequently developed to explore and exploit the language's potential.

Finally, we argue that this case illustrates the performative power of programming languages. Prolog encoded many of the dominant ideas in symbolic artificial intelligence at the time. When the language migrated to the architectural design context, it brought these ideas with it, enabling new conceptual and technical approaches in computer-aided design. In doing so, Prolog served as a vehicle for transferring epistemological assumptions and methodological frameworks from AI research into the field of architecture.

Thursday, December 18, 16:15-16:45
Simone Martini, University of Bologna, Italy

A case study in the design of programming constructs: Exception handling

"Good general theory does not search for the maximum generality, but for the right generality"
[S. Mac Lane, *Categories for the Working Mathematician*, 1971].

A construct in a high-level programming language is a carefully crafted "*abstraction*" over lower-level operations. But how did we get to a particular construct, with its linguistic details *and* semantics? Programming languages are full of attempts that never stood the test of time, while a few of these proposals "sticked" and became a common feature of most languages of wider use. The reasons for their success may be many—simplicity, generality, beauty, efficient implementation, etc.—and of course, they vary from construct to construct. Among the many examples, we want to tell the history of exception handling, where simplicity and "design orthogonality" emerge over time, preferring simpler to more general—and apparently more sensible—semantics. What makes exception handling particularly interesting is that we have both the search for a syntax *and* the competition between two different semantics—the *resumption* and the *termination* model—in which the less general termination model, which is also the furthest from the *interrupts* of the physical machine, will win. We will tell this story as a CS example of Mac Lane's maxim quoted above.

Exception handling constructs in modern languages are all similar to Python's

```
try: <block> except: <handler>
```

coupling in a single construct both the `<block>` where an exception may happen and its `<handler>`. All modern languages follow a termination semantics: if an exception occurs during the evaluation of the `<block>` , that block is terminated at the exception point and the execution is transferred to the `<handler>`; upon termination of the `<handler>`, the execution continues with whatever follows the whole construct. In a resumption semantics, instead, when an exception occurs, the `<block>` would be suspended, waiting for the `<handler>` to terminate, at which point its execution would be "resumed" at the point where the exception occurred.

After Lisp's `errorset` function, the first language to include an exception handling construct is PL/I, with its ON `<exception>` command [6], which defines and activates a handler for a specific `<exception>`. From the moment such a command is evaluated, if the `<exception>` occurs, the handler is executed, under a resume semantics by default. PL/I is inspired by interrupts and their handlers at the system level, and thus the resumption model is the most natural, despite the complexity of its implementation with arbitrarily nested function calls. We are still well before the structured programming wave [2]—no mention is made of a protected block restricting where an exception may activate the handler, and thus the activation applies to the entire program text yet to be executed after the ON command.

Coupling a block to a handler in a lexically scoped construct is the proposal of John Goodenough's theoretical paper [3], which also discusses termination and resumption semantics in detail. While in PL/I exceptions are still a way of handling error conditions (on I/O, or on arithmetic), here we see exception handling also as a proper programming technique, "as a means of conveniently interleaving actions belonging to different levels of abstraction." Goodenough also proposes that a function definition declares all the exceptions it *may* raise, a requirement which will materialize in its strong form only in Java's (much discussed) *checked* exceptions (via Modula-3 procedure signatures [1], which, however, raise only run-time, and not compile-time, errors.) Exceptions as communications between computations are the subject of Roy Levin's PhD thesis at CMU [4], which discusses why and when one should prefer a specific exception handling mechanism to a more general communication construct. All these theoretical contributions coalesced into the design of CLU [5], which argues its choice of the termination model as a "tradeoff between the expressive power of the exception handling mechanism and its complexity." Resumption semantics remained an option for a decade, especially in special contexts, like Xerox PARC's Mesa language (which

will influence the design of Modula-3), until C++, which adopted termination after years of discussion [7, page 390ff], at which point exception handling constructs reached a stable form in most high-level programming languages.

[1] L. Cardelli, J. Donahue, L. Glassman, et al. Modula-3 language definition. *SIGPLAN Not.*, 27(8):15–42, August 1992.

[2] O.-J. Dahl, E. W. Dijkstra, and C. A.-R. Hoare. *Structured programming*. Academic Press, London, 1972.

[3] J. B. Goodenough. Exception handling: issues and a proposed notation. *Commun. ACM*, 18(12):683–696, December 1975.

[4] R. Levin. *Program Structures for Exceptional Condition Handling*. PhD thesis, CMU, CS Dpt, Pittsburgh, 1977.

[5] B. H. Liskov and A. Snyder. Exception handling in CLU. *IEEE Trans. on Software Engineering*, SE-5(6):546–558, 1979.

[6] G. Radin and H. P. Rogoway. NPL: highlights of a new programming language. *Commun. ACM*, 8(1):9–17, January 1965.

[7] B. Stroustroup. *The Design and Evolution of C++*. Addison-Wesley, 1994.

Thursday, December 18, 17:00-17:30
Jad Kadan, Tel Aviv University and Forter Inc., Israel

Simultaneous code: Multiple invention and the early culture of programming automation, 1947–1952

The phenomenon of simultaneous invention stands out in the early history of computing. Between 1947 and 1952, teams working an ocean apart (Grace Hopper's group in the United States and Maurice Wilkes's group at Cambridge, UK) devised strikingly similar tools for programming automation: catalogued subroutine libraries, symbolic input formats, and relocation schemes that spared programmers the manual work of hand-coding addresses. This paper asks why these parallel systems surfaced almost at once, and what their convergence reveals about the emerging culture of code.

Drawing on laboratory reports, internal memos, surviving program decks, and oral-history interviews, the study reconstructs the technical trajectories of two notable projects: Hopper's evolving subroutine work on the Mark I/III and UNIVAC, culminating in the A-0 compiler; and Wilkes, Wheeler, and Gill's EDSAC "Initial Orders," "Synthetic Orders," and expanding subroutine catalogue. Close comparison shows how both groups, facing escalating program length and limited debugging time, responded by carving routine mathematical procedures into reusable blocks and by devising symbolic call-words that an automated linker would translate into machine addresses on the fly. Yet the solutions were not carbon copies: Hopper's team framed automation as a way to let "the programmer return to being a mathematician," while Wilkes stressed disciplined documentation and library governance. Their divergent motives mattered less than the shared bottlenecks of early stored-program machines: tiny memories, slow I/O, and users impatient for results.

Placing these cases side by side supports two claims. First, multiple invention was in the center of early software practice. Once symbolic addressing met modifiable memory, the idea of a relocatable subroutine library was now feasible and separate groups could reach it without direct contact. Second, treating automation as simultaneous rather than singular shifts the spotlight from individual ingenuity to collective and iterative craft. Program libraries grew through local tinkering, peer exchange, and informal borrowing; innovation travelled by internal memoranda and conference discussions as much as by published papers.

The argument extends beyond chronology. It feeds into philosophical debates on what counts as invention when a technology is intrinsically collaborative, and into cultural histories that trace how coding norms take root. Recognising the simultaneous origins of programming automation complicates linear origin stories and anchors early software in its sociotechnical matrices: wartime ballistics labs, university computing services, early commercial goals, and the material limits of mercury delay lines and magnetic drums.

By reframing these early tools as products of collective simultaneity, the paper offers a fresh vantage on the birth of software engineering and invites further reflection on how scarcity, community, and shared imagination shape the very idea of a "program."

Thursday, December 18, 17:30-18:00
Tomas Petricek, Charles University, Prague, Czechia

Cultures of programming. The development of programming concepts and methodologies

Programming has multiple diverse origins ranging from logic and electrical engineering to art and psychology [1]. Different *cultures of programming* that are rooted in those disciplines continue shaping programming to this day. They clash over the nature of programming, but also productively contribute to crucial programming concepts [2]. They also remain at the core of many past and ongoing disagreements about programming [3]. The answer to what is a correct program and how to create one differs dramatically when we see programs as mathematical entities, engineered socio-technical systems or media for assisting human thought.

In the talk, I will identify five different cultures of programming and explore their way of thinking about programming through a number of historical case studies. The *mathematical culture* sees programming as a mathematical activity and advocates the use of formal methods. The *hacker culture* sees programming as tinkering and emphasizes practical skills of an individual [4].The *managerial culture* aims to solve problems via organizational structures and processes. The *engineering culture* emphasizes methodologies and tools that help individuals and teams achieve satisfactory results. Finally, the *humanistic culture* focuses on creativity, interaction and broader implications of computing.

To show how different cultures of programming think about programming, how they interact, how they clash and how they contribute towards the development of new programming concepts and methodologies, I look at six historical case studies. They cover the mathematization of programming [5], the history of interactive programming tools and systems, the rise of software engineering, and the development of the programming language notions of types [6] and object-oriented programming.

The notion of *cultures of programming* presented in the talk is an abstraction that provides a lens for looking at the history of programming. As any such abstraction, it omits important details, but it also provides a revealing perspective, showing that programming has historically benefited from its fundamentally pluralistic nature.

[1] Documented e.g., by Michael Mahoney (*Histories of Computing*, Harvard University Press, 2011).

[2] For example, the birth of the notion of a programming language, studied by Nofre et al. (When technology became language, *Technology and culture*, 2014), can be seen as arising at the intersection of the hacker culture (developing custom order codes), managerial culture (seeking universal language for portability) and mathematical culture (developing early formal language descriptions).

[3] In the verification debate documented by MacKenzie (*Mechanizing proof*, MIT Press, 2004), the mathematical culture views programs as mathematical entities that admit formal proofs, whereas the engineering culture emphasizes their material properties and effect in the physical world.

[4] I provide a structure for folklore notions such as the hacker culture, which became a household term thanks to Levy (*Hackers: Heroes of the Computer Revolution*, Anchor/ Doubleday, 1984), but has also been subject to critical examination, e.g., by Haigh (*When Hackers Were Heroes*, CACM, 2021).

[5] Building on the work of Priestley (A science of operations, Springer, 2011), Astarte (*Formalising meaning*, Newcastle University, 2019) and others.

[6] Building on the work of Kell (*In search of types*, Onward! 2014), Martini (*Several types of types in programming languages*, Springer, 2015) and others.

Friday, December 19, 10:45-11:15

Chirine Laghjichi, LIPN (Laboratoire d'informatique de Paris Nord), Université Sorbonne Paris Nord

Physical computation and the condition of locality

This talk will question wether the condition of locality that is stated in different models of computation can redefine a notion of physical step. The condition of locality setting a restriction on the transition of a state to the next, thus accounting for the elementary step of computation, was initially justified by Turing (Turing, 1936) by the limitation of the human sensory apparatus. It was later formally stated by Gandy (Gandy, 1980) and sets a bound on the neighborhood of a state, representing the information needed for one step to be executed. If locality expresses the effectiveness of computation when it is formalized for digital models such as the Turing machine, is there a similar condition for analog models of computation that would stand for a physical account of computation? The difficulty we are facing in considering the condition of locality as a principle for physical computation pertains to the fact that, in its digital version, it seems to rely inherently on the finiteness of the representation of the information. Thus, if locality is defined for symbolic models of computation such as the Turing machine, setting a bound on the squares that the head is observing in order to compute the next state, how can an analog model of computation, computing over the reals, set such a bound? Furthermore, locality participates in the understanding of computation as a discrete procedure executed in a step-wise fashion. Can it play a similar role for analog models of computation computing in continuous time?

In order to answer such a question, we are analyzing the attempts at formalizing what would state as a condition of locality that applies to a procedure computing over reals, where information is continuously variable and the finiteness of information not presupposed. We claim that the condition of locality is getting a more precise definition in analog models due to the fact that it relies on the necessity of formalizing the finiteness or the approximation of the information treated. Thus, if we take analog models of computation as accounting for the physical account, we can observe that the notion of step depends on the notion of information or its approximation, depending on the model at stake. This contrasts the locality for symbolic accounts that don't need this requirement since the finite representation of information is presupposed in the formalization.

The first part of the talk will go further in the distinction proposed by Soare (Soare, 1996) between the mechanistic and the recursive accounts of computation. We link the mechanistic account to the explicit statement of the condition of locality that first appeared in the Turing machine. It has an equivalent in Gurevich's ASM enunciated as the postulate of bounded exploration. We will see that, although it is justified for reasons pertaining to the concrete execution of the system, it does have a formal expression that relies on restrictions of finiteness and participates in determining an elementary step of computation.

The second part of the talk will introduce analog models of computation and see to which extent they are representative of physical computation. We will use the specific example of BSS models, taking values of its registers in R, simulating any kind of dynamic system. We will show that they exhibit a condition of locality that is central, looking at how it might be expressed in its formalization.

In the third and last part of the talk, we are going to compare the two notions of locality for analog and digital models of computation. We will do so by looking at the example of the analog ASMs, an attempt based on Gurevich's axiomatization, to account also for analog computation. We will show that more focus is made on the precision given to the bounded exploration, that we took in the first part as being the equivalent of the condition of locality, this time reinforced in the formalization of the physical account.

References

[1] Blum, Lenore, Mike Shub, et Steve Smale. 1989. "On a Theory of Computation and Complexity over the Real Numbers: Np-Completeness, Recursive Functions and Universal Machines". *Bulletin of the American Mathematical Society*, January

[2] Bournez, Olivier, Nachum Dershowitz, et Evgenia Falkovich. 2012. ”Towards an Axiomatization of Simple Analog Algorithms”. In *Theory and Applications of Models of Computation*, edited by Manindra Agrawal, S. Barry Cooper, et Angsheng Li, 7287: 525-36, Springer.
[3] Gandy, Robin. 1980. ”Church’s Thesis and Principles for Mechanisms”. In *Studies in Logic and the Foundations of Mathematics*, edited by Jon Barwise, H. Jerome Keisler, et Kenneth Kunen, 101:123-48. *The Kleene Symposium*. Elsevier.
[4] Gurevich, Yuri. 2000. ”Sequential abstract-state machines capture sequential algorithms”. *ACM Trans. Comput. Logic* 1 (1): 77-111.
[5] Soare, Robert I. 1996. ”Computability and Recursion”. *Bulletin of Symbolic Logic* 2 (3): 284-321. https://doi.org/10.2307/420992.
[6] Turing, A M. 1936. ”On computable numbers with an application to the Entscheidungsproblem”.

Friday, December 19, 11:15-11:45

Ben Gershon and **Oron Shagrir**, Hebrew University, Jerusalem, Israel

The mathematical objection to artificial (machine) intelligence

In a series of lectures and papers between 1947 and 1952, Alan Turing developed the idea of machine intelligence, or what we now call artificial intelligence. In some of these works, Turing raises and replies to the mathematical objection. At the heart of the objection is the "mathematical result," which "proves a disability of machines to which the human intellect is not subject" (Turing 1950: 451). The "disability of machines" refers to famous undecidability results—problems that cannot be decided by means of an algorithm or (assuming the Church- Turing Thesis) a universal Turing machine.

In this paper we address two questions. We first ask why Turing was concerned about the mathematical objection. After all, even if some humans surpass machines in their mathematical abilities, this by itself does not undermine the project of machine intelligence. Our answer is that the mathematical objection raises a dilemma with respect to Turing's core claims about machine intelligence and forces him to relinquish at least one of them. The two claims are:

1. Digital computers have the ability to pass the Turing Test.
2. The Turing Test captures (machine) intelligence—any significant difference in intelligence will manifest itself in behavior in a way that would be captured by the Turing Test.

Machine intelligence follows from these two claims. If digital computers pass the Turing test, and if the Turing test captures intelligence, then digital computers have the ability to exhibit intelligence.

The mathematical objection, however, challenges this argument. Assume the "mathematical result," which says that there is a significant mind/machine difference. We can now ask: Does the Turing test capture this difference?

- If YES—then the machine does not pass the Turing Test and so claim 1 is falsified.
- If NO—then the Turing Test misses a significant difference in intelligence and so claim 2 is falsified.

One way or another, the mathematical objection goes, the argument for machine intelligence fails (as one of the premises must be false) and so the route to machine intelligence is blocked.

The second question concerns Turing's reply to the mathematical objection. As it appears, Turing does not choose to deny that the human intellect can exceed the undecidable results. Turing (1936, 1939, 1940) in fact says that human mathematicians can excel beyond undecidability (Piccinini 2003; Copeland and Shagrir 2013; Sieg 2013). So what *is* Turing's reply? Developing a proposal by Piccinini (2003) and Copeland and Shagrir (2013), and built on textual evidence, we show that Turing rejects the mathematical result because he thinks that some machines—*enhanced machines*—can exceed these limitative results too. Enhanced machines are, roughly, machines that are allowed to make mistakes, learn from experience that includes interaction with the physical and social environment, which might also lead to the modification of the rules of operations. The machine also includes a random element that Turing mentions quite frequently when discussing machine learning. More generally, Turing thinks that if we want to build intelligent machines, we better build enhanced machines.

In the last part of the paper we inquire whether enhanced machines can exceed the limitative results and perform non-computable functions (or solve undecidable problems).

Friday, December 19, 12:00-12:30
Alexander Gschwendtner, University of Vienna, Austria

From representation to generation: The epistemic logic of latent space

As generative AI systems increasingly shape what we see and understand, *latent space* – i.e., the abstract, high-dimensional structure underlying these models – has become a central, if largely invisible, site of epistemic activity. Rather than understanding these spaces representationally, as some form of compressed encodings of learned features, I argue that they function modally and constructively: as epistemic infrastructures that enable the exploration for possible configurations, relations and outputs. Drawing on non-representational philosophies of scientific modeling (Knuuttila 2011, Morgan and Morrison 1999, Suárez 2004), I suggest these models gain epistemic traction through what they *do* – not what they depict: Models function by mediating between theoretical and empirical domains to enable inference and manipulation, not by accurately 'mirroring reality'.

This epistemic logic of latent space has precedent: Freud's (1913) conception of latent content in the interpretation of dreams (hidden, symbolic meaning requiring inference) already articulated inference based on transformation, not direct depiction. Latent spaces in generative models (unintuitive, unobservable variables requiring interpretation) inherit this logic, rearticulating it in mathematical form: structuring abstract geometries through which relationships become navigable by inference and recomposition, not resemblance.

By employing learned, task-specific compression rather than extracting all preexisting features, outputs of generative models don't necessarily mirror 'reality' as humans perceive it. They open up an approach where novel forms, meanings and inferences can be drawn from learned data, becoming a formal environment for 'expanding modal space': enabling the exploration of "what-if" scenarios and revealing configurations not directly derivable from theory (Sjölin-Wirling 2021, Rice 2025). In this context, the model's balance between compression, perceptual relevance, and modelability (RDT) (Dielemann 2025) underscores how latent spaces are optimized for epistemically generative manipulation instead of faithful representation. The result is an epistemic stance that privileges synthetic construction over correspondence.

This reorientation has philosophical stakes beyond AI or machine learning: When researchers fine-tune models, artists explore generative possibilities through vector arithmetic in latent space, or scientists use AI for drug design (Schneider et al. 2020), materials engineering (Merchant et al 2023), and protein generation (Jumper et al 2021), they are engaging with this latent infrastructure. These applications demonstrate latent space manipulation as epistemic practice – navigating abstract geometries to generate novel compositions, discover unconventional synthesis pathways, or design materials with desired properties. It compels us to reconsider what models are for: not depicting a fixed external world, but creating epistemic artifacts that allow us to navigate, test, and rethink its possibilities. Latent spaces function thereby as non-representational, generative parts of models that actively produce knowledge rather than passively storing it. Their outputs are synthetic propositions: epistemic possibilities derived from learned patterns rather than directly observed realities.

Far from being a technical side-effect then, latent space constitutes a new kind of epistemic architecture that aligns with broader shifts in philosophy of science from representational to instrumental, modal, and constructive understandings of models. Analyzing latent space through this lens contributes to an emerging view of computational models as modal epistemic instruments, as tools for the generation of knowledge.

Friday, December 19, 12:30-13:00
Giora Hon, University of Haifa, Israel

Who will watch the watchmen (*Quis custodiet ipsos custodes*)? Von Neumann and Turing on computational error

At the famous Hixon meeting in September 1948 at Caltech, von Neumann responded in the discussion of his presentation, "The General and Logical Theory of Automata", to a query concerning his claim that a machine can be constructed so as to contain safeguards against errors and provision for correcting errors when they occur. The question was posed whether a mechanism in which all innumerable contingencies have been foreseen, and the corresponding corrective measures build in, is actually conceivable? Von Neumann responded:

> An artificial machine may well be provided with organs which recognize and correct errors automatically. In fact, almost every well-planned machine contains some organs whose function is to do just this…. Furthermore, if any particular machine is given, it is always possible to construct a second machine which "watches" the first one, and which senses and possibly even corrects its errors. The trouble is, however, that now the second machine's errors are unchecked, that is, *quis custodiet ipsos custodes*? Building a third, a fourth, etc., machine for second order, third order, etc., checking merely shifts the problem. In addition, the primary and the secondary machine will, together, make more errors than the first one alone, since they have more components.

Who then indeed will watch the watchmen?

Both von Neumann and Turing were engaged intensely, right from the outset of their work on automatic computing machines, with the problem of reliability and the occurrence of error in computation.

Turing, on his part, in his "Computing Machinery and Intelligence" of 1950, made strong claims about infallible machine and proceeded to distinguish between "error of functioning" and "error of conclusion". The ides here is to direct attention to the sources of error.

> Errors of functioning are due to some mechanical or electrical fault which causes the machine to behave otherwise than it was designed to do. In philosophical discussions one likes to ignore the possibility of such error; one is therefore discussing "abstract machines." These abstract machines are mathematical fictions rather than physical objects. By definition they are incapable of errors of functioning. In this sense we can truly say that "machines can never make mistakes." Errors of conclusion can only arise when some meaning is attached to the output signals from the machine. The machine might, for instance, type out mathematical equations, or sentences in English. When a false proposition is typed we say that the machine has committed an error of conclusion.

In my paper I will juxtapose the contrasting view of the protagonists' concept of error in automatic computation, and will pay special attention to the contexts in which these claims were made and the respective commitments of von Neumann and Turing.