HaPoC 2013

Maarten Bullynck, Jean-Baptiste Joinet

HaPoC 2013

History and Philosophy of Computing

October 28th-31st 2013
Paris France

Ecole Normale Supérieure

Organised by
Centre Cavaillès – CIRPHLES

Conference Aims and Scope

Computing has pervaded our everyday lives. Beginning with the democratisation of the computer in the 1980ies to the ubiquitous embedded computing entities of today, the spread of computing on all levels of our societies and our lives, both private and professional, does not seem to have come to a halt yet. Despite this, the knowledge and know-how on computing is for most hidden behind intricate imprints of microprocessors, layers of programming, processed by specialised knowledge and finally made acquaintable by a variety of user-friendly interfaces. Given the impact of computing, not only on our scientific work, but also on our mundane experience, it is a task of utmost importance to make the nature and characteristics of computing available for reflection, not only by experts, but by a broad community of researchers and thinkers.

Historically, the computer is born from the complex convergence of many traditions: Scientific calculating machines, business machines, electrical circuit design, war related research on communication systems, etc. The computer's appearance almost immediately sparked off the regimentation of an important body of mathematical and logical results (of which many were essentially an outgrowth of Hilbert's metamathematical programme) into parts of what was to become (theoretical) computer science. Later on, a similar process happened in the development of conceptual approaches to programming. Moreover, the computer has had and is having a considerable influence on all sciences and has made it possible to put quite some new research questions on the agenda, often resuscitating older questions, e.g. research on algorithms in mathematics. This multiform origin of computing (informatique) is still very much a living part of the discipline and can still be the basis of a meeting ground for philosophers, logicians, historians, mathematicians, computer scientists, programmers, sociologists, artists, etc. to talk and reflect about computing. This conference wants to bring all these researchers together to stimulate a dialogue of ideas and to facilitate an interplay of methods and concepts to help form and nurture the emerging and exciting field of history and philosophy of computing. On a more systematic level, the topics of the conference include (but are not restricted to):

1. History and Philosophy of Computability (E.g., interpretation and actuality of the Church- Turing thesis; other models of computability ...)
2. History and Philosophy of Computing and Programming (E.g., Mod-

els of computing; logical and mathematical foundations of computer science; information theory; classes of programming languages...)

3. Epistemology of the use of computing in the sciences (E.g., simulation vs. modelisation; computer-assisted proofs; exploration...)

4. History and Philosophy of the Computer (E.g., from calculating machines to the future of the computer; abstract architectures...)

5. Social, ethical and paedagogical issues of Computing (E.g., education of informatics; algorithms and copyright; internet, culture and society; web philosophy...)

6. Art and Informatics

This conference History and Philosophy of Computing (HaPoC 2013) is the follow-up conference to the first History and Philosophy of Computing that was organised in Gent (Belgium) November 2011. Its success and the enthusiasm it generated among its participants vouch for a successful second edition. The participation of many institutions in the organisation of the conference, the presence of many personalities in the field to the programme committee and the list of invited speakers testify of the general import that is expected from the conference. The planning of several special issues is a guarantee that the results of the conference will be translated as a series of articles that will serve as references for the future of the field.

Chairs

Maarten Bullynck (Paris 8 & SPHERE UMR 7219 CNRS)
Jean-Baptiste Joinet (Lyon 3, IRPhil EA 4187 & Centre Cavaillès, Cirphles USR 3308 CNRS)

Local Organising Committee

Dominique Hoarau, secrétariat Cirphles (ENS)

Iro Bartzia (Paris 8)
Florent Franchette (docteur, Paris 1)
Baptiste Mélès (post-doc, Archives Poincaré Nancy)
Alberto Naibo (doctorant, Paris 1)
Maël Pégny (doctorant, Paris 1)
Patrice Pereira (doctorant, Paris 8)
Mattia Petrolo (doctorant, Paris 7)

Laurent Ungerer (Ecole Nationale Supérieure des Arts Décoratifs)

Scientific and Programme Committee

Gerard Alberts (Amsterdam)
Sergei Artemov (New York)
Gérard Berry (Paris)
Cristian Calude (Auckland)
Martin Campbell-Kelly (Warwick)
Leo Corry (Tel Aviv)
Liesbeth De Mol (Lille)
Marie-José Durand-Richard (Paris)
Helena Durnova (Brno)
Jean-Louis Giavitto (Paris)
Marie Hicks (Chicago)
Benedikt Loewe (Amsterdam)
Simone Martini (Bologna)
Pierre-Eric Mounier-Kuhn (Paris)
Gualtiero Piccinini (Missouri)
Giuseppe Primiero ()
William Rapaport (Buffalo)
Sonja Smets (Groningen)
Julian Rohrhuber (Düsseldorf)
Raul Rojas (Berlin)
Jérôme Ségal (Vienna)
Wilfried Sieg (Carnegie Mellon)
Raymond Turner (Essex)

Organization and Support

HaPoC 2013 is supported by

Centre Cavaillès

Cirphles (USR 3308 CNRS, ENS)

Ecole Normale Supérieure

PSL (Paris, Sciences et Lettres)

SPHERE (CNRS, UMR 7219)

Ecole Nationale Supérieure des Arts Décoratifs

INRIA (Institut National de Recherche en Informatique et Automatique)

GDR 3398 Histoire des mathématiques

Equipe "Sciences et Technologies de la Musique et du Son" (UMR 9912 IRCAM/CNRS)

CNRS (Centre National de Recherche Scientifique)

Schedule

Monday 28th October

9h00-9h45 Inscription + coffee break

9h45-10h00 Welcome and Introduction

10h00-11h00 Invited Talk: Janet Abbate, *Is Computer Science a Science? A Half-Century Debate*

Contributed Session 1: Fundamental Questions

11h00-11h30 Marcello Pelillo , Teresa Scantamburlo and Viola Schiaffonati . *Computer Science between Science and Technology: A Red Herring?*

11h30-12h00 Edgar Daylight. *On the Difficulties of Writing about the History of Computing*

12h00-12h30 Valérie Schafer, Francesca Musiani and Benjamin Thierry. *Is Networking computing ?*

12h30-14h30 Lunch

14h30-15h30 Invited Talk: Gilles Dowek, *Informatics and the classification of sciences*

Contributed Session 2: Epistemological, ethical and paedagogical aspects of Computing

15h30-16h00 Julie Jebeile. *Verification & Validation of Computer Simulations: A Philosophical Analysis*

16h00-16h30 Luca Gasparri and Jacopo Tagliabue. *We don't want to miss a thing. Objecthood in a digital universe*

16h30-17h00 Coffee break

17h00-17h30 Harry Halpin. *The Logical and Philosophical Foundations of the Open World Assumption*

17h30-18h00 Elisabetta Mori. *Tomás Maldonado and the Sign System for Olivetti ELEA 9003*

18h00-18h30 Fabio Gadducci and Giovanni Cignoni. *A syllabus for the Fifties. Teaching computer science on the first Italian computers.*

18h30-19h00 Yann Secq. *40 years of computer science PhD in Lille University*

Tuesday 29th October

9h00-10h00 Invited Talk: Walter Dean, *Algorithms and ontology*

10h00-10h30 Coffee break

Contributed Session 3: History and Philosophy of the Computer

10h30-11h00 Robert Dennhardt. *The term 'digital' and 'Digital Computer' by George Robert Stibitz 1942*

11h00-11h30 Mark Priestley. *Computers and obedience: defining machine autonomy in the 1940s*

11h30-12h00 Camille Paloque-Berges, Haud Guegen and Claire Scopsi. *What network computing does to communication. A retrospective analysis of early debates confronting and inventing online communication ethics*

12h00-12h30 Wolfgang Brand. *Weaving the Net: Places of Computing and their Political, Social and Technical Interconnections*

12h30-14h30 Lunch

14h30-15h30 Invited Talk: Jean-Yves Girard, *Logic revisited through informatics*

Contributed Session 4: History and Philosophy of Computability

15h30-16h00 Guido Gherardi. *The applications of Turing (in)computability to classical mathematics*

16h00-16h30 Teresa Numerico. *Cybernetics, control and Big data*

16h30-17h00 Coffee break

17h00-17h30 Mate Szabo. *Kalmár's Argument Against the Plausibility of Church's Thesis*

17h30-18h00 James Power. *Exploring Thue's 1914 paper on the transformation of strings according to given rules*

18h00-18h30 Christopher Porter. *Von Mises, Church, and the Birth of Algorithmic Randomness*

18h30-19h00 Mirko Tavosanis. *Back to the (Libraries of the) Future*

Wednesday 30th October

9h00-10h00 Invited Talk: Nathan Ensmenger, *The multiple meanings of a flowchart: visual representations of complexity in computer programming*

10h00-10h30 Coffee break

Contributed Session 5: History and Philosophy of Programming

10h30-11h00 Federico Gobbo and Marco Benini. *The epistemology of programming language paradigms*

11h00-11h30 Baptiste Mélès. *Philology of Programming Languages*

11h30-12h00 Andrea Valle and Alessandro Mazzei. *Towards a Semiotic Framework for Programming Languages*

12h00-12h30 Alexandre Hocquet. *RTFM! Scientific modeling and the generification of software*

12h30-14h30 Lunch

14h30-15h00 Introduction and welcome at Ecole Nationale Supérieure des Arts Décoratifs

15h00-16h00 Invited Talk: Margit Rosen, *A Technological Difference, Not a Difference Of Method. On the Notion of Programming in the Arts of the 1960s*

Contributed Session 6: Computing and the Arts : historical and conceptual issues

16h00-16h30 Mario Verdicchio. *Can Computing in Art Renew the Debate on Art?*

16h30-17h00 Joanna Walewska. *Computer art or art of computing? Early debates revisited.*

17h00-17h15 Break

17h15-17h45 Jean-Marc Wolff. *An historical perspective on informatics language and music composition*

17h45-18h45 Performance by IRCAM-members Moreno Andreata, Gérard Assayag and Jean-Louis Giavitto, "Sciences et Technologies de la Musique et du Son" (SMTS, UMR 9912 IRCAM / CNRS) in the Amphithéâtre

19h00-22h00 Performance 'The Great C' organised by ENSAD-Lab, curators: Antoine Schmitt, Jean-Jacques Birgé (in the Amphi Bachelier). In parallel there is the walking dinner (in the 'Rotonde')

Thurday 31st October

9h00-10h00 Invited Talk: Barry Cooper, *Types of Thinking*

10h00-10h30 Coffee break

Contributed Session 7: History and Philosophy of Networking

10h30-11h00 Federica Russo and Silvia Crafa. *Causality in concurrent systems*

11h00-11h30 Felice Cardone. *Computers as communication machines – Highlights of a forgotten program*

11h30-12h30 Invited Talk: Bernard Chazelle, *Computation as a conceptual tool for modern science*

12h30-13h00 Closing Remarks and Goodbye

13h00-15h00 Lunch

List of Invited Talks

Is Computer Science a Science? A Half-Century Debate

Janet Abbate (abbate@computer.org)
Science and Technology in Society, Virginia Tech, USA

From the 1950s to the present, computer scientists have debated whether their field is a science or not. Many have lamented that they are not recognized as scientists by other scientists, funding agencies, or the public. Why have computer scientists debated this question for so long, and what might their answers tell us about the nature of computing and the nature of science itself? This paper surveys how the meaning of computer science has changed over time and connects these evolving definitions with social and material contexts including changes in technology, the institutionalization of computer science within universities and professional societies, industry demand for computer experts, and funding opportunities. I then focus on debates over the scientific status of computing. Which elements of computer science are identified as "scientific," and what do these claims reveal about how science itself is perceived and valued? And what is at stake in this debate? The paper will focus mainly on US but will incorporate comparative examples from Europe and the USSR.

Types of Thinking

Barry Cooper (pmt6sbc@maths.leeds.ac.uk)
School of Mathematics, University of Leeds, United Kingdom

Early work on artificial intelligence was predicated on the assumption that the digital computer might emulate what had been thought of as distinctively human modes of thinking. Turing himself at Hanslope Park in 1944 was quoted as saying he was "building a brain". On the other hand, there is a long history of recognition of different kinds of thinking, described variously as: head versus heart; logical versus intuitive; scientific versus artistic; left brain versus right brain, etc. Jacques Hadamard's essay on 'Psychology of Invention in the Mathematical Field', drawing extensively on the thinking of Henri Poincare (as outlined in his famous lecture on mathematical invention,

in Paris in 1908), came just after Turing's 1939 paper bringing mathematical logic to bear on the relationship between ingenuity and intuition.

The theme running through all this work, and through later work of Turing and others, is that the differences between kinds of thinking emerging from observation, introspection and the mathematics are puzzlingly real. And that these might be the basis for an essentially cooperative conjunction of modes. Related to this were earlier attempts – for instance those of the so-called 'British emergentists' of the 1920s – to place this in a broader scientific context. ÂăWork on the relationship between descriptive and more clearly computational characterisations of phenomena can be traced through such early work as that of Hans Reichenbach and more recent contributions, including that of Reichenbach's one-time student Hilary Putnam.

In the more hermetic recursion theoretic context, we have Stephen Kleene and his successors developing notions of computation over objects of different type. And recently we have Luciano Floridi's book on the "Philosophy of Information", describing an information-based approach to structural realism - a key ingredient of which is the notion of 'Level of Abstraction'. In this talk we try to draw together these various strands, outlining the formative thinking and historical background, and point the way towards a clarifying mathematics.

Computation as a conceptual tool for modern science

Bernard Chazelle (chazelle@CS.Princeton.EDU)
Princeton University, USA

It was the singular genius of Alan Turing to capture the essence of computing with a machine: to compute, logicians were told to become engineers. Today, the favor is being returned. With the indispensability of the computer firmly established among scientists, we are now witnessing the rise of "computational thinking." This more ambitious phase of Turing's visionary program will see the algorithmic paradigm get woven ever more tightly into the fabric of modern science, especially biology. This (self-contained) talk will discuss this phenomenon with concrete examples.

Algorithms and ontology

Walter Dean (W.H.Dean@warwick.ac.uk)
Department of Philosophy, University of Warwick, Coventry, United Kingdom

The broad goal of this paper is to bring to the attention of philosophers of mathematics and computation the concept of *algorithm* (e.g Euclid's algorithm, Strassen's algorithm, the Gröbner basis algorithm) as it is studied in contemporary theoretical computer science, and at the same time address several foundational questions about the role this notion plays in mathematical practice. A variety of considerations such as the need to prove correctness and provide running time analyses suggest that algorithms ought to be assimilated to mathematical objects such as models of computation or recursion schemes – a view which is embodied by the well-known proposals of Yiannis Moschovakis and Yuri Gurevich. I will suggest instead that a variety of considerations grounded in complexity theory and algorithmic analysis serve as in principle obstacles to making such an identification.

Informatics and the classification of sciences

Gilles Dowek (gilles.dowek@inria.fr)
INRIA, Paris, France

The apparition of a new science poses different kinds of questions: What is this new science and how is it different from others? Can this new science be integrated in existing classifications of the sciences before its appearance, or do these classifications have to be modified? How does the apparition change our vision of what the sciences are as a whole? In this talk I will try to present some elements of an answer for the specific case of computer science.

The multiple meanings of a flowchart: visual representations of complexity in computer programming

Nathan Ensmenger (nensmeng@indiana.edu)
School Of Informatics And Computing, Indiana University, USA

A well-written computer program is, in theory at least, self-documenting; that is to say, the computer code itself contains its own complete written

specification. And yet despite the computer scientist Donald Knuth's famous claim that computer programs, like literature, were meant to be read by humans as much as machines, for the most part computer programs are too arcane and idiosyncratic to serve as a useful function as a design document.

From the very earliest days of electronic computing, "flow diagrams" (later "flowcharts") have been used to represent the conceptual structure of complex software systems. In much of the literature on software development, the flowchart serves as the central design document around which systems analysts, computer programmers, and end-users communicate, negotiate, and represent complexity. And yet the meaning of any particular flowchart was often highly contested, and the apparent specificity of such design documents rarely reflected reality. In fact, some of the first software "packages" (commercial applications that could be purchased off-the-shelf) were used to reverse-engineer the flowchart specification from already developed computer code. In other words, the implementation of many software systems actually preceded their own design! Drawing on the sociological concept of the boundary object, I will explore the material culture of software development, with a particular focus on the ways in which flowcharts served as political artifacts within the emerging communities of practices of computer programming.

Logic revisited through informatics

Jean-Yves Girard (girard@iml.univ-mrs.fr)
CNRS, CIRM Luminy, France

The climateric date in recent logic is 1931 : incompleteness puts an end to XIXth century, scientist logic. However, XXth century logic, which starts with Gentzen, Herbrand and Kolmogorov was still relying on XIXth century schemes: Hilbert's formalism and Russell's logicism, until computer science, through its multiple connections with logic, provided a fresh grid.

Indeed, the logical universe can be enlightened by three basic oppositions :

1. What is an answer (implicit/explicit) ?
2. What is a question (formatted/informal) ?
3. What conveys certainty (epidictic/apodictic) ?

A Technological Difference, Not a Difference Of Method
On the Notion of Programming in the Arts of the 1960s.

Margit Rosen (margitrosen@gmx.de)
ZKM Karlsruhe, Germany

When electronic computers caught the attention of artists and art theorists in the 1960ies, the idea of a "rational", "scientific" or "programmable" art was in the focus of interest. The technology seemed to offer a way out of the much criticized arbitrariness of post war abstract art and its alleged social irrelevance. This paper explores the use of the concept of "programming" both on the base of historical theoretical texts as well as of the actual handling of computers by the artists. The consideration of the use of self-built analog or hybrid devices to which artists referred to as "computers" not only allows for deepening the discussion of the idea "programming" in the artistic context, but for adressing the methodological problem of approaching so called "computer art".

Special Session on "Computing and the Arts"

The afternoon of Wednesday, 30th October, HaPoC 2013 will take place at the Amphithéâtre Bachelier in the Ecole Nationale Superieure des Arts Decoratifs, 31 rue d'Ulm, a few steps away from the Ecole Normale Superieure (ENS, rue d'Ulm 45) where the conference takes place.

This afternoon features

- an invited talk by Margit Rosen (ZKM Karlsruhe)
- a Contributed Session on 'Computing and the Arts : historical and conceptual issues'
- a performance by members of the research group 'Sciences et Technologies de la Musique et du Son' (SMTS, UMR 9912 IRCAM / CNRS), Moreno Andreata, Gérard Assayag and Jean-Louis Giavitto, in the Amphithéâtre of the Ecole Nationale Supérieure des Arts Décoratifs (17h45-18h45)
- and a performance/exhibition with the title **The Great C**, organised by the ENSAD-lab under the direction of Antoine SchmittAntoine Schmitt and Jean-Jacques Birgé (19h-22h).

Synchronously to the exhibition and performance in the Amphithéâtre, a walking dinner will also take place at la Rotonde, immediately next to the Amphithéâtre.

Abstracts of Contributed Talks

Weaving the Net: Places of Computing and their Political, Social and Technical Interconnections

Wolfgang Brand (wolfgang.brand@ims.uni-stuttgart.de)
Universität Stuttgart, Historisches Institut, Abt. für Geschichte der Naturwissenschaften und Technik, Stuttgart, Germany

Today, most people carry with them an enormous amount of computing power in their mobile devices. Historically, substantial amounts of computing power were concentrated in computer centres where large calculation machines consumed a lot of resources and had - compared with today - rather limited capabilities.

The first computers were solitary machines located in laboratories and operated mostly by their creators. During the 1950s and 1960s computer centres were established to house these machines and specially trained operators were running these devices filling the gap between the constructors and the users of these artefacts. But what processes and boundary conditions formed these computing centres? What kind of people worked at these places of computing and how did these centres interconnect or not?

This historical case study will examine the development of high-performance computer centres in the German state of Baden-Württemberg from the 1960s to the end of the last century. It will cover the political, social and technical aspects of their development. The developments in Baden-Württemberg were chosen as the field of study because right from the beginning this region spearheaded the quest for high-performance computing in Germany. Created from two rivalling states after the Second World War, Baden-Württemberg has a political culture where a delicate balance of power between its two constituents has to be maintained. This is reflected in the allocation of resources and the formation of (super)- computing centres in this state. Starting in the late 1970s a sometimes painful process started where the existing scientific computing centres began to collaborate more closely and a hierarchical structure regarding the size and technical potentials of these centres emerged. This structure was reflected by the interconnection, both on a structural and a technical level (communication networks), between these centres which led to a two tier structure of high performance computing in the whole of Germany with a small group of three centres at the top.

Baden-Württemberg was one of the first places in Germany where large-scale computers were deployed in scientific computation. After John Argyris, the co-inventor of the Finite Elements Method (FEM), came to Stuttgart University in the late 1950s substantial efforts were made to establish a state-of-the-art computing environment in Stuttgart. These efforts were echoed by the other universities such as Karlsruhe, Tübingen and Heidelberg who established their own computing centres. Especially Karlsruhe University, a university with a strong engineering tradition, tried to position itself for a long time as a potential host of the most powerful computing centre in Baden-Württemberg. In the late 1970s, when a political decision by the state government was prepared to buy one of the first Cray 1 supercomputers available, an open rivalry between the computing centres broke out. This conflict, also fueled by the main protagonists, threatened to paralyze the whole processes to establish top computer capabilities.

During summer 1981 it became evident that an arrangement had to be found to end (or at least channel) the rivalry between Stuttgart and Karlsruhe to make any progress. This resulted in the "Konstanzer Seefrieden" (peace treaty at the Lake of Konstanz) where the leaders of the computing centres agreed to meet on a regular basis (four times a year) to discuss the relevant topics in a "peaceful atmosphere". The small town of Achern was selected as a neutral place to meet and the participants named themselves "Acherner Kreis" (Achern circle). Finally, the computer centre at Stuttgart University was selected as the main supercomputer centre in Baden-Württemberg and all other university computer centres had to limit their ambitions to somewhat less powerful computers. This strategic decision also promoted the early interconnection between the computing centres and in 1987 the BelWü, the first non-US TCP/IP-based scientific network became operational.

The contribution is mainly based on materials never used before in a historical study. A major part of the material is based on oral interviews with those who participated in these developments. First-hand accounts are given on the development of high performance computer centres, their structure and interconnections.

This study is part of the author's PhD thesis project on the history of high performance computing and the contributions of the Stuttgart and Karlsruhe areas at the Department for the History of the Natural Sciences and Technology at the University of Stuttgart.

References

Held, Wilhelm et. al.: Geschichte der Zusammenarbeit der Rechenzentren in Forschung und Lehre vom Betrieb der ersten Rechner bis zur heutigen Kommunikation und Informationsverarbeitung, ZKI, Manuscript 2009

Original material from the archives of Stuttgart and Karlsruhe University

Oral interviews with participants and witnesses of these developments

Computers as communication machines – Highlights of a forgotten program

Felice Cardone (felice@di.unito.it)
Dipartimento di Informatica, Università di Torino, Italy

Issuing a purchase order in a large company, transferring rights by delivery of a promissory note, making a motion in a deliberative assembly under Robert's Rules of Order, are example of coordination patterns among human activities that rely upon disciplined communication among participants playing formally defined roles. Although much effort has been invested of late into exploiting computers in these situations, the conventional metaphor of computer as a calculating machine fails to provide the rationale behind such effort.

Motivated by this observation, we rediscover an approach to the conceptual role of computers pioneered by Carl Adam Petri (1926-2010) and Anatol W. Holt (1927-2010), who regarded the computer as a "general medium for strictly organized information flow" (Petri 1976), a "communication machine" (Holt 1975) in the context of a specific organized activity. Our reconstruction is mostly based on the work of Holt, whose lifelong research interests were the construction of a theory explaining the role of computers in the coordination of human activities, and the realization of computational artifacts in support of organized activity (Holt 1997).

Through his acquaintance with Gregory Bateson, Anatol Holt inter- acted with cyberneticians and systemically oriented thinkers[1] and, viewing a system as a means to establish certain relations of communication among a set of role players,[2] he set the foundations for

[1] See (Bateson 1972). Holt was one of the main characters of that conference which included as participants Barry Commoner, Warren McCulloch and Gordon Pask among others.

[2] For reasons of space we omit full reference to unpublished material. This commu-

> a theory about the mechanical aspects of communication – i.e., those aspects that have to do with the rules, insofar as these can be formalized, which define the relations among a set of communicating parts [. . .] We regard the words 'organization' and 'system' as referring to such bodies of rules.

Around the end of the 1960s, Holt's ideas contributed to spread sys- temic notions like concurrency, conflict and causality, and the formalism of Petri nets (Holt & Commoner 1970), influencing in particular the research on asynchronous hardware, where the absence of a global clock makes it necessary to achieve coordination among components by means of suitable signaling schemes.

The vision of computer as a communication machine essentially situated within the context of human organizations gave rise to an unconvential research program. Communication here does not just mean, as in Shannon's seminal work, "reproducing at one point either exactly or approximately a message selected at another point": the reception via fax of a perfect reproduction of a 100 dollar bill does not count as a successful money transfer. In place of a theory of signal transmission like classical communication theory we rather need a formal pragmatics providing the conceptual tools for the analysis and synthesis of patterns of communication. While applying communication mechanics to the design of electronic coordination environments, Holt introduced examples of disciplines to be imposed on message-handling capabilities within a computer-based information system, like delegation of authority, addressing of messages and their identification and cancellation. Petri (1976) compiled a list of such communication disciplines classifying the functions of computer as a general medium for strictly organized information flow,

> disciplines of a science of communication yet to be created, and disciplines in the sense of keeping to a set of rules to be followed if communication is to be successful (Petri 2001).

While only a few fragments of such a new science of communication have been developed, work toward this goal has provided valuable insights. On the one hand, we have the beginnings of a theory of coordination based on notions like role, activity, state, behavior and interaction, exploiting a graphical formalism for expressing coordination patterns (Grimes et al. 1983).

nication mechanics later developed into a theory of coor- dination whose formalization instantiated the elements of the bipartite ontology of Petri nets as roles and activities (Grimes et al. 1983) or, later, as bodies and operations (Holt 1985, 1997).

On the other hand, considerations on the relations between discrete and continuous models stimulated by this formalism have led to a refor- mulation of (Dedekind-)continuity via non-transitive similarity relations, like concurrency and indistinguishability by measurement, admitting continuous structures of countable size (Petri 1982, (C5) p. 985).

It is our conviction that a systematic account of these insights will contribute to revive a research program which is not just of historical interest.

References

Bateson, M. C. (1972), Our Own Metaphor. A Personal Account of a Conference on the Effects of Conscious Purpose on Human Adaptation, Smithsonian Institution.

Grimes, J. D., Holt, A. W. & Ramsey, H. R. (1983), 'Coordination sys- tem technology as the basis for a programming environment', Electrical Communications 57(4), 301–314.

Holt, A. W. (1975), Formal methods in system analysis., in Shaw, B., ed., 'Computers and the Educated Individual', University of Newcastle upon Tyne, pp. 135–179. http://www.ncl.ac.uk/computing/about/history/seminars/.

Holt, A. W. (1985), Coordination technology and Petri nets, in G. Rozenberg, ed., 'Advances in Petri Nets', Vol. 222 of Lecture Notes in Com- puter Science, Springer-Verlag, pp. 278–296.

Holt, A. W. (1997), Organized Activity and its Support by Computer, Kluwer.

Holt, A. W. & Commoner, F. (1970), Events and conditions, in J. B. Dennis, ed., 'Record of the Project MAC conference on concurrent systems and parallel computation', ACM, New York, NY, USA, pp. 3–52.

Petri, C. A. (1976), Communication disciplines, in B. Shaw, ed., 'Computing System Design', University of Newcastle upon Tyne, pp. 171–183. http://www.ncl.ac.uk/computing/about/history/seminars/.

Petri, C. A. (1982), 'State transition structures in physics and computa- tion', International Journal of Theoretical Physics 21(12), 979–992.

Petri, C. A. (2001), 'Cultural aspects of net theory', Soft Computing 5, 141–145.

On the Difficulties of Writing about the History of Computing

Edgar Daylight (egdaylight@yahoo.com)
Eindhoven University of Technology, Netherlands

My mentor has advised me not to use technological concepts, like 'program', 'compiler', and 'universal Turing machine', as subjects of my sentences. Instead, I should use historical actors. For example, I should not write

> During the 1950s, a universal Turing machine became widely viewed as a conceptual abstraction of a computer.

Instead, I should write

> By 1955, Gorn viewed a universal Turing machine as a conceptual ab- straction of a loop controlled computer.

If I stick to sentences of the first kind, my exposition will, at best, capture a de- velopment of ideas that is detached from the people who shaped the technology in the first place. As a result, my readership, and myself included, won't realize that a universal Turing machine had different meanings for different actors, nor will it become apparent that the meaning of a universal Turing machine changed over time for each individual actor. Gorn, for example, viewed a universal Tur- ing machine quite differently in 1955 than a year or two prior.

Sentences of the first kind can lead to more pitfalls. They sometimes go along with expositions in which one line of thought dominates the entire narrative. For example, if I choose as subject matter the connection between Turing's 1936 universal machine and modern computers, then it becomes tempting to view the history of computing in terms of those few people who grasped that connection early on. Turing already saw it around 1946 but this observation alone does not make him an influential historical actor. By following my mentor's advice, the historian loses the temptation to paint the history of computing as a beautiful road from logic to practice; that is, as a stream from Turing's 1936 paper to the stored-program computer. The historian will fail, and this is for the good, in explaining every technological advancement in terms of Turing machines.

The late Mahoney warned his colleagues not to fall into the trap of viewing everything with the same glasses. "The computer", Mahoney said specifically, "is not one thing but many different things, and the same holds true

of com- puting" Mahoney 2011, p.25–26). Instead of taking a 'machine-centered' view of history, we should examine actual computing practices, including programming practices – as Haigh elaborated in the introduction of Mahoney's collected works (Mahoney 2011, p.5,8). In my words, then, the multitude of programming styles should come to the fore. Terms like 'general purpose' and 'universal' had different meanings for different historical actors and I thus take Mahoney's challenge to mean that we, historians, need to handle each and every term with great care. Using 'general purpose' as a synonym for 'Turing universal' is only justified if it conforms to the historical context.

Unfortunately, Mahoney 'never took up his own invitation', said Haigh. Men like McCarthy, Scott, Strachey, Turing, and von Neumann appear in almost every chapter in Mahoney's collected works, but, as Haigh continued,

> [W]e never really learn who these people are, how their ideas were formed and around what kind of practice, what their broader agendas were, or whether anything they said had a direct influence on the subsequent de- velopment of programming work. (Mahoney 2011, p.8)

Mahoney did occasionally provide insights about the aforementioned men. Co- incidentally or not, the rare passages in which he did also comply with my mentor's writing style, as the following excerpt illustrates.

> Christopher Strachey had learned about the λ-calculus from Roger Pen- rose in 1958 and had engaged Peter J. Landin to look into its application to formal semantics. (Mahoney 2011, p.172)

These words inform the reader about some historical actors, their social net- works, and research objectives. However, instead of using historical actors as subjects, Mahoney often used technological concepts. The following paragraph, published in 2002, illustrates Mahoney's typical writing style:

> The operating system became the master choreographer in an ever-more complex dance of processes, coordinating them to move tightly among one another, singly and in groups, yet without colliding. The task required the development of sophisticated techniques of exception-handling and dynamic data management, but the possibility of carrying it out at all rested ultimately on the computer's capacity to rewrite its own tape. (Mahoney 2011, p.83–84, my emphasis)

Mahoney chose 'the operating system' and 'the task' at hand as subject matters. And, although his prose is exceptional, it is not clear to me who abode with 'the task' and who did not. Or did Mahoney intend to convey that ev- eryone shared a common technological point of view? Moreover, did everyone explain their technology in terms of a tape? a 'Turing tape'? Or was this solely Mahoney's personal interpretation?

In my talk I will elaborately compare the writing styles of Mahoney (2011), Ceruzzi (2012), Copeland (2012), and Priestley (2011), with the sole purpose of obtaining a better understanding on how to effectively document the history of computing.

References

P.E. Ceruzzi. Computing: A Concise History. MIT Press, 2012.

B. Jack Copeland. Turing: Pioneer of the Information Age. Oxford University Press, 2012.

M.S. Mahoney. Histories of Computing. Harvard University Press, 2011.

M. Priestley. A Science of Operations: Machines, Logic and the Invention of Programming. Springer, 2011.

The term 'digital' and 'Digital Computer' by George Robert Stibitz 1942

Robert Dennhardt (robert.dennhardt@web.de)
Platanus School, Berlin, Germany

Why do the terms digital and therefore digital computer seem to be so problematic? In all literature dealing with computer history, there is a certain indifference concerning the differently scaled function descriptions and terminologies of individual parts or the entire computer and its analogue, discreet or digital properties. For example, in Williams Aspray's *Computing Before Computers* (1990) Paul Ceruzzi compared the terms analogue and digital by juxtaposing the technical term 'analogue' with another one to initiate a technological or epistemological opposition. "Indeed, Atanasoff was the first to use the word 'analogue' to describe that type of computer (Atanasoff-Berry-Com- puter ABC, 1939); 'digital' and 'digital computer' was first used by George Robert Stibitz in 1942." 1 So far, no author was able to state a direct source.

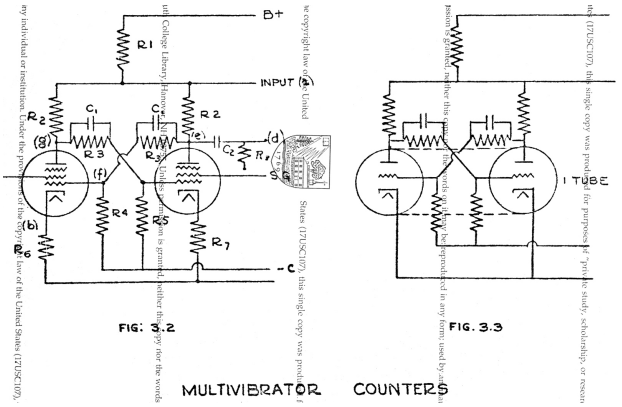
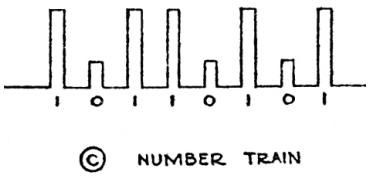


Figure 1. Stibitz does not mention the origin of the term multivibrator counter anywhere. The function 'counter' means one-bit shift register. The astable multivibrator by Abraham and Bloch from 1917 therefore remains unquoted. The only difference between multivibrator and flip-flop merely exists in the selection of the feedback elements, meaning capacitors with the multivibrator, resistors with the flip-flop or bistable trigger element, as well as one capacitor and resistor each with the monostable trigger element or monoflop. The capacitors here merely serve to accelerate switching. The multivibrator counter therefore corresponds to Turner's Kallirotron.

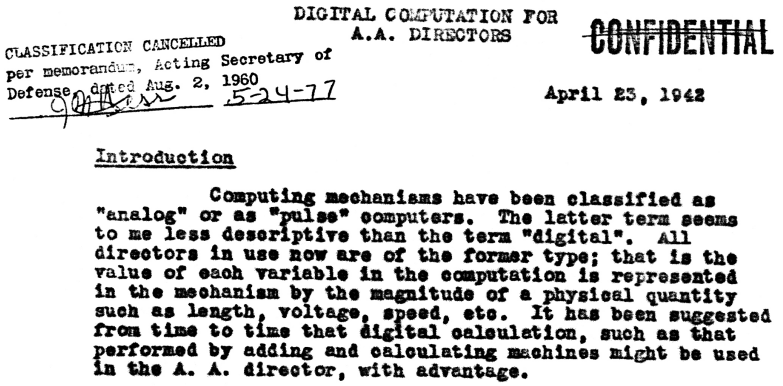
Illustration 1 originates from the closing report *Report on electronic predictors for anti-aircraft fire control* about a computer-assisted anti-aircraft system from Stibitz' estate dating back to the beginning of April 1942, in which he pointed out the fundamental advantages of merely having to program and compute two discreet voltage levels contrary to the exclusive programming of analogue voltage signals or many discreet voltage levels so far.



© NUMBER TRAIN

Meticulousness, intuition and coincidence joined forces when âĂŞ based solely on a re- mark in Friedrich Wilhelm Hagemeyer's publication *Die Entstehung von Information- skonzepten in der Nachrichtentechnik* (1979) – I assumed that the author could be in possession of a copy of the decisive memorandum Digital Computation For A.(nti) A.(ircraft) Directors dating back to the end of 1942. Once I pointed this out to Hagemeyer, he was

surprised to own such a treasure. It may well be that he was the only one to have made a copy in June 1977. He handed me a copy of the cover page (fig. 2) and let me copy the important passage on page three.



DIGITAL COMPUTATION FOR
A.A. DIRECTORS

~~CONFIDENTIAL~~

CLASSIFICATION CANCELLED
per memorandum, Acting Secretary of
Defense dated Aug. 2, 1960
5-24-77

April 23, 1942

Introduction

Computing mechanisms have been classified as "analog" or as "pulse" computers. The latter term seems to me less descriptive than the term "digital". All directors in use now are of the former type; that is the value of each variable in the computation is represented in the mechanism by the magnitude of a physical quantity such as length, voltage, speed, etc. It has been suggested from time to time that digital calculation, such as that performed by adding and calculating machines might be used in the A. A. director, with advantage.

Figure 2. "Digital computers introduce a consideration not found in kinematic analogue computers, namely, the ordering of computation steps in time [number train, see fig. above text] In a vague sense, therefore, digital computation is dynamic in character, but so far as I know no theory exists. [...]"

Based on this document, we can safely assume that the term 'digital computer' was invented by Stibitz approximately three years before the completion of the first electronic digital computers COLOSSUS and ENIAC. A sufficient prerequisite for the unambiguous property of the digital computer can only be defined negatively with Stibitz' elaborations: A digital computer does not have any analogue properties regarding interfaces and computing in any systemic scaling.

With the prehistory of the computer analysed in my work and Stibitz' documents, it is possible to word a larger media history context as genesis of digital saving parallel to the genesis of binary discreet pulsing, because the flip-flop was the genesis of this synchronisation technology as well. This means that the clear dynamic addressing of any number of binary flip-flop memories was first made possible by the synchronisation of buffered computing data and/or the computing of ones and zeros from partially external program memories and internal main memories. John von Neumann's document *First Draft Report on the EDVAC* dating back to the middle of 1945 is also a part of this genesis. The calculator EDVAC, developed at the end of 1945, emerged from the ENIAC dating back to the beginning of

1945. ENIAC computed fully electronically in several parallel working units, meaning it had no punch tape command memory. For new programs, however, many tube connections of the main memory had to be interchanged manually. Although the COLOSSUS (end of 1943) was the first computer to compute binary electronic, it had to be partially re-cabled for new computing programs. In addition, certain data was provided on punch tape for both computers.

To summarise this essay, the answers to both initial questions concerning the origin of the flip-flop and the problems with the term 'digital' are inseparably linked with each other inasmuch as it was not possible to detect the problem with the term 'digital' or only word it very vaguely, because its technical and discursive genealogy lay in the dark. As a kind of circle, this in turn was one of the reasons why there was no technical or scientific need to follow up on the assumption that Stibitz may have left behind a document containing basic statements about the technological status and technological mode of the digital with reference to the digital computer. It was possible to clear up the origin of the term 'digital computer' and expand the content of the term 'digital' or word it more precisely. Still, with respect to common linguistic usage and scientific jargon, the following questions still remain virulent, because analogue is not the opposite of digital: Why do we have electronic music on the one hand, but digital media on the other hand? And if there are electronic media, what is the possible difference? A lot of scientific, technological and cultural-historical studies could be based on this.

References

P. E. Ceruzzi, 'Electronic Calculators', in: W. Aspray, *Computing Before Computers*, Iowa State University Press, Ames 1990.

G. R. Stibitz, Digital Computation For A. A. Directors, 23. April 1942, p. 3. One copy is in possession of Friedrich Wilhelm Hagemeyer. The original is probably lying in some box at the Library of Congress in Washington. It is, however, not among Stibitz' estate, administered by the Dartmouth College Library in Hanover, New Hampshire, USA.

A syllabus for the Fifties. Teaching computer science on the first Italian computers.

Fabio Gadducci (gadducci@di.unipi.it)
Department of Informatics & Museum of the Computing Instruments of the University of Pisa, Italy

Giovanni Cignoni (giovanni@di.unipi.it)
Department of Informatics & Museum of the Computing Instruments of the University of Pisa, Italy

The earliest master degrees for computer science have been established by the mid- Sixties, with ample discussions on which should be the syllabi for their courses. At the same time, many teaching activities had already been established by the simple need of providing the reuired expertise for the machines built since the Forties. Concerning e.g. Italy, we know of the network established by Olivetti for allowing technicians to program and interact with its ELEA series. Less information is available e.g. for the courses taught at an academic level. At least for the early Italian computers, though, we are in a privileged position, having available both the original notes and the current reminiscences of the protagonists of these courses at the University of Pisa in the late Fifties. The CEP project is one of the founding myths of Italian computer science. Hosted by the University of Pisa, supported by the counties of Livorno, Lucca, and Pisa and sponsored financially and technically by Olivetti, in the years 1955-1961 it produced the first Italian computer, called Macchina Ridotta (MR, 1957) and on the basis of such an accomplishment it later delivered the long-running Calcolatrice Elettronica Pisana (the eponymous CEP, 1961). The project can be considered the seed from which the first Italian master degree on computer science (1969) will develop. Moreover, it implicitly contributed to the development of the first Italian transistored computer, one of the earliest in Europe: the ELEA 9003 by Olivetti. The most recent and accurate reconstruction so far of the CEP project is (Cignoni and Gaducci 2012).

The original designers apparently undervalued the costs and required times for software development on their machines (Cignoni and Gaducci 2012). However, since the beginning they clearly understood the need of teaching the basics of programming, and its relevance in the spread of the soon-to-be-build machine. What was designed and tested in the project was to be immediately used for the benefit of the transfer of knowledge. Indeed, the activity report of July 1956 also documents the educational activities

that were carried out during the first semester for a dozen of graduating engineering students.

The 1956 course was held between March and May. It was split in a few modules, taught by the MR designers, according to their role in the project. While the engineers supported by Olivetti, Giuseppe Cecchini and Sergio Sibani, focussed on the electronic design, the physicists working for the University, Alfono Caracciolo and Elio Fabri, focussed on the architectural and programming aspects, respectively. Concerning the latter, we have an undated technical report containing the notes of the module, most likely the first written text devoted to teaching computer science in Italy (Fabri 1958). Eight lessons, transcribed and published by Centro Studi Calcolatrici Elettroniche (CSCE), the managing body of the CEP project (fully subsided by the University of Pisa) (Fabri 1958). The title "Introduction to the programming of a calculating machine" clearly shows its applied approach, and its tight connection with the actual programming of the first version of MR.

The first lesson indeed introduces the basic of Von Neumann architecture, even if it explicitly states that each word in 18 bits long (as in MR). The second lists the commands available for MR, and illustrates some simple programs. The third and fourth explains in details the (assembly) program for calculating the max of a series of numbers. Most interestingly, lessons 5 through 7 illustrate the use of flow diagrams for program specification, in particular for cycles of possibly undetermined length. The final lesson discusses instead some practical issues: the need of an entry device (in this case, the tape) and of a stored program for the boot, activated by a Manual Control Desk. After a detour on the difference between permanent and temporary memories, the lesson is rounded up by the introduction of the concept of subprogram: its usefulness for programming, and the way to store them.

For the Fifties, among the surviving documents of the CEP projects (scattared among the Pisa archives) we found traces of just another course, held during the academic year 1958/59. This was a fully-fledged course, though, even if only the professor register (briefly annotating the contents of each lesson) was preserved (Böhm 1958/59). However, it is remarkable the range of the topics the course deals with in its 45 lessons. A most important aspect is the professor himself: Corrado Böhm, one of the founding father of Italian theoretical computer science, who was by then collaborating with CSCE through Centro Calcoli Numerici of the Technical University in Milan.

The course started in December, 2, 1958, and by then the MR was already disassembled, in order to be cannibalized for the 1961 machine: with the

exception of Caracciolo, its designers had already left the project. This reason likely contributed to the less hands-on structure of the course: after laying down the basics of information theory, it moves to binary arithmetic and logics. It offers an introduction on function interpolations, but mostly importantly, at least for the modern viewers, the central lessons deal with Moore automata and Turing machine, as well as adopting a "simplified calculator" ("calcolatrice semplificata") for illustrating some programs.

These two "syllabi" illustrate the attention to the state of the art. For example, the seminal work by Moore on Gedanken-experiments was published just in 1956. Indeed, the course by Böhm (more than the one by Fabri, clearly with a more hands-on attitude) can be considered as one of the first witnesses of a certain idea of the completeness of training: attention to the technological challenges of the moment, but based on a solid conceptual basis to provide the ability to continue to study. One of the basis of the soon-to-be established undergraduate degree in Computer Science.

References

Giovanni A. Cignoni, Fabio Gadducci. Rediscovering the Very First Italian Digital Computer. Proceedings of 3rd IEEE History of Electro-technology Conference (HISTELCON), IEEE, 2012.

Elio Fabri. Appunti delle lezioni di "Introduzione alla programmazione di una calcolatrice elettronica" (raccolte e redatte da Luigina Bosman Fabri). Technical Report CSCE 35, 1958. Available `http://hmr.di.unipi.it/DocCEP/1958\_NI1-35-Appunti.pdf`

Corrado Böhm. Registro delle lezioni di calcoli numerici e grafici dettate dal Sig. Prof. C. Böhm. University of Pisa, Faculty of Science, academic year 1958/59. Available at `http://hmr.di.unipi.it/DocCEP/1959\_RegistroBoehm.pdf`

We don't want to miss a thing. Objecthood in a digital universe

Luca Gasparri (`lcgasparri@gmail.com`)
Vita-Salute San Raffaele University, Italy
Jacopo Tagliabue (`tagliabue.jacopo@gmail.com`)
Vita-Salute San Raffaele University, Italy

The notion of *object* is a basic and fundamental concept in philosophy of language, formal semantics, cognitive psychology, ontology: objects are the

references of our names, the values of our variables, the content of our propositional attitudes, the building blocks of our universe.

In the psychological literature on vision, the label *object perception* is used to refer to four different kinds of processes (Mather 2006, Goldstein 2010):

i) the integration and segregation of elements in the visual input according to the detection of salient contours;

ii) the assignment of shape and structure to some of those elements;

iii) the categorization and the semantic labeling of shaped contours;

iv) the concentration of attention on shaped contours.

Let us focus on (i). How do observers determine that a region of space is occupied by an *object*, i.e., by a complex entity that 'can be introduced into discussion by means of a singular, definitely identifying substantival expression' (Strawson 1959: 137)? Evidence suggests that integration, segregation and object grouping are early visual processes that occur independently from semantic categorization and attentional location. Research has also shown that the likelihood that a set of spatial regions will be grouped together, segregated from other spatial regions, and taken to host an object, is dependent on a regular set of parameters, such as similarity, proximity, convexity, common movement, and smallness relative to the ground area (Peterson 2001). However, our most basic intuitions about the requirements that a given portion of space has to satisfy in order to be represented as occupied by an object are often unclear and still at the heart of a vivid philosophical and scientific debate. Thus, a question arises naturally: can we provide a neutral, precise metrics to analyze objecthood judgments and devise a taxonomy of different conceptions of objecthood within a unified framework?

This work presents a preliminary proposal in this direction by exploiting digital, computational devices known as cellular automata (hence CA). In particular, we shall use CA as "toy universes" in which to investigate the complex interplay between cognitive processes and conceptual issues in object detection. CA are paradigmatic examples of complex systems (Mainzer, Chua 2012): although their structure is incredibly simple, they are capable of amazingly complex behavior (Ilachinski 2001, Wolfram 2002). CA can be simply implemented in a PC as n-dimensional digital universes satisfying the following constraints:

CA_1) space is discrete and made of fundamental atoms (*cells*);

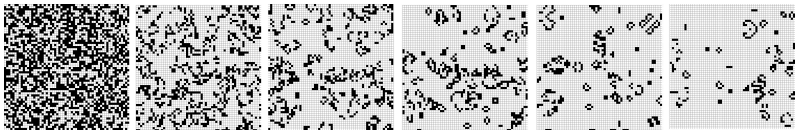
CA_2) time is discrete and made of *instants*;

CA_3) cells have one property at each instant of time (a *state*), chosen from a finite set of possibilities;

CA_4) at each instant of time, each cell updates its state through a *transition function* defined over the states of its *neighboring* cells (e.g., 'If all my neighbors were in state S_1 at time t, assume state S_2 at time $t + 1$').

CA are entirely defined by (and entirely reducible to) these four elements: everything that happens in such worlds is completely determined by the arrangement of cells' states in the space-time, which in turn is completely determined by the initial conditions of the system (the "Big Bang") and the local transition function.

As vividly shown by the snapshots collected below from the CA *Life* (Berlekamp, Conway, Guy 1982), CA support the existence of "complex patterns", "emergent objects moving through the universe", "causal histories". However, the complexity of the behavior of the universe can always be reduced to the local application of a simple deterministic rule: we can vary *all* the basic and emergent features of the universe just by changing the arrangement of cells' states in the lattice.



For the purposes of the present work, CA are interesting because they can be studied at three different levels:

1) *a "logico-philosophicus" level*: the ontology of the universe allows for a very neat, transparent formalization using standard tools from mereotopology and formal ontology (Casati, Varzi 1999). Philosophical accounts of objecthood (Laycock 2010) can be easily tested, compared and evaluated in the proposed framework.

2) *a cognitive level*: the amazing emerging behavior of CA provides the basis for rigorous and repeatable experiments on objecthood judgments in human subjects. In particular, given the computational nature of the universe, the experimenter is free to vary at will any condition in the universe and measure the cognitive and linguistic responses in the subjects.

3) *a computational level*: thanks to their digital structure, CA allow for precise calculations on the amount of *information* generated by the universe evolution: following a suggestion by Daniel Dennett (Dennett 1991, Ross

2000), we can calculate an "informational" notion of objecthood, according to which objects are statistically informative patterns in the space-time.

Our goal is to provide a rigorous theoretical and methodological framework for an investigation of objecthood judgments in CA universes. We shall argue that a study of the algorithmic conditions under which groups of cells in a CA universe are assigned the status of object promises to provide interesting insights on the nature of the psychological processes of objecthood assignment, as well as a well-behaved and regimented basis to investigate the philosophical status of the notion of *object*.

A crucial part of the proposal is to combine conceptual analysis with experimental data. By exploiting the digital nature of CA we can test (for example):

i) the dependence of objecthood judgments on the underlying CA rule;

ii) the top-down inference from emergent objects to the updating rule;

iii) the difference and interplay between intuitive assignments of objecthood and the "informational" account of objecthood.

Given the centrality of *things* in human everyday cognition, conceptual inquiry and scientific enterprise, a thorough bottom-up investigation of the philosophical, computational, psychological foundations of objecthood judgments is needed to shed some new light on this important issue.

References

Berlekamp, E., Conway J., Guy R., 1982, *Winning Ways for Your Mathematical Plays (Vol. 2)*, London: Academic Press.

Casati, R., Varzi A., 1999, *Parts and Places*, Cambridge, MA: MIT Press.

Dennett, D., 1991, 'Real Patterns', *Journal of Philosophy*, 88: 27-51.

Goldstein, E. B., 2010, *Sensation and Perception*, 8^{th} ed., Belmont, CA: Wadsworth.

Ilachinski, A., 2001, *Cellular Automata*, Singapore: World Scientific Publishin

Laycock, H., 2010, "Object", *The Stanford Encyclopedia of Philosophy (Fall 2010 Edition)*, E. N.Zalta (ed.), http://plato.stanford.edu/archives/fall2010/entries/object/.

Mainzer, K., Chua L., 2012, *The Universe as Automaton. From Simplicity and Symmetry to Complexity*, Heidelberg: Springer.

Mather, G., 2006, *Foundations of Perception*, New York, NY: Psychology Press.

Peterson, M. A., 2001, 'Object Perception', in E. B. Goldstein (ed.), *Blackwell Handbook of Sensation and Perception*, Oxford: Blackwell, pp. 169-203.

Ross, D., 2000, 'Rainforest Realism: A Dennettian Theory of Existence', in D. Ross, A. Brook, D. Thompson (eds.), *Dennett's Philosophy A Comprehensive Assessment*, Cambridge, MA: MIT Press, pp. 147-168.

Strawson, P. F., 1959, *Individuals: An Essay in Descriptive Metaphysics*, London: Methuen.

Wolfram, S., 2002, *A New Kind of Science*, Champaign, IL: Wolfram Media.

The applications of Turing (in)computability to classical mathematics

Guido Gherardi (guido.gherardi@unibw.de)
Inst. für Informatik, Universität der Bundeswehr, München, Germany

It is well known that Turing's celebrated work on abstract computing machines (Turing 1936) provides a foundation for computability theory alternative to Church's λ-calculus. But if the latter is scarcely known outside the computer science com- munity, Turing machines benefit of an ample degree of familiarity even by the larger audience of non-experts as one of the most important scientific achievements of the last century. A primary reason for their great appeal lies in their simple definition: no preliminary mathematical knowledge is required to understand what they are and what they do.

But 77 years after their introduction, what role do Turing machines really play today in science?

While Church's λ-calculus, or functional programming languages in general, constitutes a concrete tool for algorithmic design, Turing machines seem to have primarily "theoretical applications". In fact their programming language is very inefficient and their hardware contains infinite components that cannot be constructed in reality. Vice versa, they simulate computational processes in such an informative way that they provide a very natural model for the solution of questions about the decidability of problems or about the spatial/temporal complexity of algorithms. This is not surprising, as these machines had been originally ideated by Turing himself to understand the notion of computability and its limits rather than to execute "actual computations".

Correspondingly, the substantial difference between λ-calculus and the Turing machine model has a deep impact in the foundations of mathematics.

Via the Curry-Howard isomorphism, λ-calculus is particularly appreciated for its successful applications in the extraction of computational content from constructive proofs. In contrast, the rejection of Markov's principle in Bishop's standard foundation for constructivism has relegated the use of Turing machines primarily to the domain of classical mathematics.

Fortunately Turing machines have found in the latter field very significant applications, although still regarded by most classical analysts as "external" objects to pure mathematics. Such old-fashioned opinion should be challenged by the results recently obtained in topology and analysis through the use of oracle Turing machines. These well-known and more sophisticated versions of Turing machines can store and manipulate incomputable objects and hence they amplify the use of the notion of Turing computability in the realm of incom- putability. For topological reasons, their tape-based model allows us to evaluate in a relatively easy way how discontinuity phenomena make some information transformation process over the real continuum incomputable, and this would be probably hardly achieved within the λ-calculus. Therefore they are nowadays commonly used in fields of computer science where incomputability play a major role, such as recursion theory or Weihrauch's approach to computable analysis (Weihrauch 2000). In particular they guarantee the existence of those real computable functions whose domains contain non computable numbers.

Several examples of their meaningful mathematical applications come from descriptive set theory, as a part of classical topology, in particular those regarding the relations between Borel measurability and Weihrauch redicibility (Brattka 2005) (for instance, Borel measurable functions can be characterized as those functions that are topologically Weihrauch reducible to the closed choice operator on the Baire space (Brattka 2012)).

Some even more significant examples concern one of the core concepts of analysis, differentiability (and could be summarized under Nies's Motto "Ran- domness = Differentiability"). Classically, by saying that a property holds for a random real $z \in [0, 1]$ we mean that the reals failing that property form a null set. But computability theory provides us with finer definitions of randomness with respect to that "naive" classical notion (and in each of such computational notions a random real is always non computable, i.e. it has a classical but not constructive existence). For instance, by a theorem of Lebesgue every nonde- creasing function $f : [0; 1] \to \mathbb{R}$ is differentiable at all reals z outside a null set depending on f . Recently, Brattka et al. have proved that a real number is computably random iff every nondecreasing computable function $f : [0, 1] \to \mathbb{R}$ is differentiable at z. This statement is really an improvement of the original Lebesgue Theorem: it formulates that

result not only through a technically well-determined notion of randomness, but the considered randomness class is also independent from the choice of f .

Already in the 70's Demuth (1975) had proved that a real z is Martin-Löf random iff every computable function of bounded variation is differentiable at z. Recently, Bienvenu et al. (2012) have worked on the classical Denjoy-Young-Sacks Theorem, according to that for every function $f : [0,1] \to \mathbb{R}$ and almost all z, either $f'(z)$ exists or the upper (lower) derivative at z diverges to ∞ (- ∞). In particular they have proved that a real z is computably random iff every computable function $f : [0,1] \to \mathbb{R}$ satisfies the Denjoy alternative at z.

Finally Pathak et al. (to appear) have re-formulated the Lebesgue Differentiation Theorem as follows: every real $z \in [0,1]^d$ is Schnorr random iff $\lim_{Q \to z} \to z \frac{1}{\lambda(Q)} \int_Q g$ exists for every Lebesgue-integrable computable function $g : [0,1]^d \to \mathbb{R}$, where Q is an open cube containing z with volume $\lambda(Q) \to 0$.

These results are remarkable, because all randomness notions involved were already available and fundamental in the literature; hence they are not new ad-hoc inventions. Therefore these theorems show that intrinsic connections between core notions of computability theory and real analysis actually exist.

References

L. Bienvenu, R. Hölzl, J. Miller, A. Nies: The Denjoy alternative for computable functions. Proc. 29th Symposium on Theoretical Aspects of Computer Science: 543-554 (2012)

V. Brattka: Effective Borel measurability and reducibility of functions. MLQ 51:19-44 (2005)

V. Brattka, M. de Brecht A. Pauly: Closed Choice and a uniform Low Basis Theorem, APAL 163:986-1008 (2012)

V. Brattka, J. Miller, A. Nies: Randomness and Differentiability. Submitted.

O. Demuth: The differentiability of constructive functions of weakly bounded variation on pseudo numbers. Comment. Math. Univ. Carolinae 17:583-599 (1975)

N. Pathak, C. Rojas, S. Simpson: Schnorr randomness and the Lebesgue Differentiation Theorem, Proc. Amer. Mathem. Society, to appear.

A. Turing: On computable numbers, with an application to the "Entscheidungsproblem", Proc. London Math. Society 42:230-265 (1936)

K. Weihrauch: Computable Analysis. Springer (2000)

The epistemology of programming language paradigms

Federico Gobbo (federico.gobbo@univaq.it)
DISIM - University of L'Aquila, Italy
Marco Benini (marco.benini@uninsubria.it)
DICOM – Department of Computer Science and Communication, University of Insubria, Varese, Italy

The history of modern computer programming languages can be traced back to the struc- tured program theorem by Böm & Jacopini (1966) and the j'accuse against the goto statement by Dijkstra (1968). In October of the same year, the conference organised by the NATO Science Committee introduced the concept of 'software engineering', and an IFIP Working Group on 'Programming Methodology' was established. As recalled by Dijkstra (2001): '[IBM] did not like the popularity of my text; it stole the term "Structured Pro- gramming" and under its auspices Harlan D. Mills trivialized the original concept to the abolishment of the goto statement.' It became evident that computer programming ought to be more solid, both theoretically and practically, also because software projects were becoming more complex, involving an increasing number of programmers, as for instance the IBM System/360 family and in particular the OS/360 (Brooks 1995).

One of the strategies adopted by computer scientists to cope with this growing complexity was to design new programming languages, at different Levels of Abstraction (LoAs). The method of LoAs–fully explained and defended in Floridi (2008)–can be used to describe programming in terms of informational organisms (inforgs): the programmer is the infor- mational agent, while the computing machinery is the artificial artifact. A software project is an infosphere, including processes and mutual relations among the inforgs directed to the same goal. Under this perspective, the source code, i.e., the observable, is the main LoA acting with the Levels of Organisations (LoOs) of the machinery, i.e., the hierarchical structure of hardware de re. Therefore, the choice of the programming language is crucial, as it determines the epistemological approach sustaining the programmers' goals, which is identified as a Level of Explanation (LoE) by Floridi (2008). Unlike LoAs and LoOs, LoEs do not really pertain to the system, rather they are an epistemological lens through which the informational agent(s) approaches the goal of programming.

Examining computational inforgs–where the artificial artifact is a Von Neumann Machine (VNM)–Gobbo & Benini (2013) propose to look at the history of modern computing in terms of information hiding. Here, the scope is narrower, the choice of the programming language in terms of LoEs being the research question; however, the concept of informa- tion hiding can be usefully applied straightforwardly. In the early days, there was only a machine-tailored assembler letting programmers write one-to-one machine-readable in- structions. Afterwards, a fundamental LoA was introduced by Backus during the design of Fortran (Backus 1978) and its implementation via a compiler, i.e., the computer program that translates the source code into machine code. In fact, he provided a formal nota- tion that became the standard to describe programming languages: the Backus-Naur Form (BNF) abstracts over the language, allowing to compute on its structures, and thus it is a new LoA of computational inforgs. The next LoA in programming has been introduced after Böm & Jacopini (1966), a result that permitted to hide the way the machine interprets the flow of control, and to change it to something which can be easily analysed mathematically. This result opened the door to the construction of a plethora of programming languages, each one adding LoAs to hide or change the behaviour of some aspect of the machine. In fact, to cope with the growing complexity of the problems throughout the history of computing (Ceruzzi 2003), computer scientists (on the theoretical side, e.g., McCarthy's Lisp) and informaticians (on the practical side, e.g., Cobol) construed languages in order to facilitate the modelling the possible solutions of a given family of problems. We can classify programming languages in few major paradigms, according to the information got hidden. Paraphrasing the three layered description of programming by Hofstadter (1979), we can consider the source code as the novel, the programmer being the novelist, while the programming language is the literary genre of the novel.

If the VNM should be step-by-step programmed, procedural languages such as C or Pascal, the direct descendants of structural programming, are apt to the goal: as underlined by White (2004) in Floridi (2004): 'most programming languages allow programs to perform actions that change the values of variables, or which have other irreversible effects (input or output, for example); we say that these actions have side-effects'.

On the contrary, if the problem is better conceived as a formal entity, the classical paradigm of mathematics can be used. In functional programming, the modelled world is described in terms of pure functions, taking as the LoE the tradition of computation as application of mathematical operations.

Another approach is by the introduction of the concept of object, which is

conceived originally on a different LoE, philosophically based on Leibniz's monads and the notions of 20th century physic and biology (Kay 1993). It's a more sophisticated world, as the problem is split into different (virtual) VNMs that communicate one to the other by messaging and changing the local state of the object.

Finally, the fourth approach completely hides the VNM under a logical theory, so to let the program forget the algorithmic details–covered by the artificial reasoner–and modelling the problem in logical terms. Prototypically, this is the strategy followed by the Prolog language, based on the Horn clauses and unification. In practice, the procedural side of programming cannot be eliminated completely (Lloyd 1984).

It is evident that each programming language envisages a 'vision of the world' which is suitable for some classes of problems. When dealing with really complex problems, which happens in most contemporary software design, rarely a single language has the right fea- tures to model the whole problem. In this paper, we wanted to contribute to clarify the epistemological statements behind the major classes of programming languages, together with their mutual relations. By providing a taxonomy, it becomes possible to implement notions from one language into another, simulating the LoAs and the features not originally present.

References

Backus, J. (1978), 'The history of FORTRAN I, II, and III', SIGPLAN Not. 13(8), 165–180.

Böhm, C. & Jacopini, G. (1966), 'Flow diagrams, turing machines and languages with only two formation rules', Communications of the ACM 9(5), 366–371.

Brooks, Jr., F. P. (1995), The mythical man-month (anniversary ed.), Addison-Wesley Long- man Publishing Co., Inc., Boston, MA, USA.

Ceruzzi, P. (2003), A history of modern computing, History of computing, MIT Press.

Dijkstra, E. W. (1968), 'Letters to the editor: go to statement considered harmful', Commun. ACM 11(3), 147–148.

Dijkstra, E. W. (2001), What led to "notes on structured programming". circulated privately. URL: `http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1308.PDF`

Floridi, L. (2008), 'The method of levels of abstraction', Minds Mach. 18, 303–329.

Floridi, L., ed. (2004), The Blackwell guide to the philosophy of computing and information, Blackwell, London.

Gobbo, F. & Benini, M. (2013), 'From ancient to modern computing: A history of information hiding', IEEE Annals of the History of Computing 99(PrePrints).

Hofstadter, D. R. (1979), Gödel, Escher, Bach: An Eternal Golden Braid, Basic Books, New York.

Kay, A. C. (1993), 'The early history of smalltalk', SIGPLAN Not. 28(3), 69–95.

Lloyd, J. W. (1984), Foundations of logic programming, Springer-Verlag New York, Inc., New York, NY, USA.

White, G. (2004), The Philosophy of Computer Languages, in Floridi (2004), chapter 18.

The Logical and Philosophical Foundations of the Open World Assumption

Harry Halpin (hhalpin@ibiblio.org)
World Wide Web Consortium/MIT, USA

There has long been a debate between procedural and logical formalisms in knowledge representation in the history of artificial intelligence, and this debate has recently returned on the Web in the form of a conflict between procedural scripting languages and the logical formalisms of the Semantic Web. The earliest work in digital knowledge representations was spear-headed by Hayes and McCarthy's attempts to formalize elements of human knowledge in first-order predicate logic, where the primary vehicle of intelligence was to be considered some form of inference (Hayes and McCarthy 1969). While many researchers took up the grand challenge in various domains, soon a large number of insidious problems were encountered in terms of the expressivity of first-order logic as exemplified by the Frame Problem as well as technical issues such as decidability. Despairing of logic, a faction of AI researchers led by Winograd championed a procedural view of intelligence that regarded the logical properties of the representation as itself irrelevant if the program could successfully solve some task given some input and output (Winograd 1972). In practice, this led to the programming language code itself as being thought of as the model for human knowledge. Thus, a chasm was opened between the scruffy practice of AI modelling and logical formalists that insisted on well-structured foundations. Although the Web lacks the grand pretensions of AI and is simply looking for a usable representational

structure for external human knowledge, current application developers are using Javascript and JSON rather than the logically well-founded formalisms of the Semantic Web such as the family of OWL description logics.

This entire debate may indeed be a red herring. As shown by the history of language development, one of the deepest findings in programming language theory given by the Curry-Howard isomorphism: procedural programming languages can, if properly designed, correspond to logical formalisms (Wadler 2000). Although there was much two-way traffic between logical proof-proving in mathematics as established by Frege and the lambda calculus of Church, it was a number of years before a correspondence was actually determined. One key contribution was the invention of the subformula property by Gentzen that allowed for the simplifications of proofs. Although the correspondence was informally noticed by Curry, the formalization between Gentzen's natural deduction and the lambda calculus was formalized only in 1969 by Howard, which leads one to think that for every kind of well-formed programming language based on the lambda calculus in theory has a corresponding logic with a proof-theoretic semantics (Wadler 2000).

The unique contribution of the Web to this history is a firm commitment of Web architects like Tim Berners-Lee to what is called the Open World Assumption (Berners-Lee 1998). Informally, in an open- ended space of information like the Web one can never assume a statement is false without direct proof. Logically, this means that statements that cannot be proven to be true cannot be assumed to be false. This assumption also has a rich lineage in computer science, as it contrasts with the Closed World Assumption that states that logically statements that cannot be proven to be true can be assumed to be false. Intuitively, the Closed World Assumption means that somehow the world can be bounded and has often been phrased as an appeal to the Law of the Excluded Middle in classical logic (Dummett 1982). For example, negation as failure is a version of the assumption where the failure for the program to prove a statement is true implies the statement is false. The Semantic Web attempts to banish this assumption from the Web in the form of new kinds of databases without this assumption, but so far the Semantic Web has failed to attract programmers as they tend to prefer programming languages that appear to be procedural.

Could the informal Open World Assumption find adequate grounding in both logic and programming via the Curry Howard Isomorphism? Indeed, it can as the removal of the Law of the Excluded Middle naturally leads one to endorse intuitionistic logic. Furthermore, the Girard-Reynolds isomorphism has been proven between second-order intuitionist predicate logic and the second-order polymorphic lambda calculus (Wadler 2001). In this way,

there is indeed a rich, if not yet connected to the Web, logical foundation for Berners-Lee's endorsement of the Open World Assumption that could save the Semantic Web, as it raises the possibility of transforming functional languages like Javascript into typed functional languages that would preserve the Open World Assumption. Unknown to Berners-Lee, the Open World Assumption also underwrites the anti-realist philosophy of Dummett, in particular Dummett's interpretation of late Wittgenstein's rule-following (Dummett 1982). This rule-following also finds a strange parallel in Berners-Lee's foundations of the Web and Semantic Web in the form of rule-following specifications by standards bodies like the IETF and W3C. On a more speculative note, could the logic of intuitionism be itself a formalization of a much deeper principle, namely our inability on metaphysical grounds to limit the number of possible ontological objects and so always leaving open the possibility of the proof-by-construction of a new object? This metaphysical stance may very well underwrite the technical generativity of the Internet. Thus, the full philosophical ramifications of the arguments over intuitionism need to be revisited in light of the Web.

References

Berners-Lee, T. (1998) What the Semantic Web can represent. `http://www.w3.org/DesignIssues/RDFnot.html`

Dummett, Michael. Realism. Synthese 52.1 (1982): 55-112.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of Artificial Intelligence. In Meltzer, B. and Michie, D., editors, Machine Intelli- gence, volume 4, pages 463–502. Edinburgh University Press.

Wadler, P. (2000). Proofs are programs: 19th century logic and 21st century computing. Dr Dobb's Journal, 313.

Wadler, P. (2001). The Girard-Reynolds Isomorphism. In International Symposium of Theoretical Aspects of Computer Software.

Winograd, T. (1972). Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. Cognitive Psychology, 3(1).

RTFM! Scientific modeling and the generification of software

Alexandre Hocquet (`alexandre.hocquet@univ-lorraine.fr`)
Université de Lorraine, France

Taking computational chemistry as an example, the aim of the present study

is to emphasize the pivotal role of software, and above all, software distribution, on the epistemological status of modeling in computational sciences.

Computational chemistry (which could be defined as the use of computer resources to solve problems in chemistry) is a scientific discipline that emerged at the same time that computers became available in research laboratories (Bolcer and Hermann 1994), and developed with the graphics terminal (Francoeur 2002), in the 70s and 80s. When computers became personal, ie a device in the research lab accessible to non specialists, an upheaval appeared in the scientific community: the scientists who were designing the molecular modeling software (the developers) were not any more the same people than those who performed the calculations (the users). Thus, in the 80s and 90s, the problem of the distribution of the software arose, and tensions appeared in the community. Should the software be shared freely? Should it be sold? Should the code source be open? Could (and should) academic institutions benefit from a "technology transfer"? Depending on what kind of licensing? The computational chemistry community was also involved with two major industries: the computer manufacturers and the pharmaceutical industry, the latter becoming a potential market for the former through modeling software (Boyd 2007). In a context of changing times (of science fundings, of market opportunities, of academic technology transfers), computational chemistry software turned from user oriented software to market oriented software. To account for the strategies, tensions, and changes over time in the community, this work explores The Computational Chemistry List (CCL), a mailing list created in 1991 to provide a discussion board to the fledgling community (Labanowski 2007). For twenty years, it has been used as an opinions forum and a platform for scientific exchange. Since its inception, through the archives of its thousands of threaded conversations, the mailing list is a valuable corpus from a Goffmanian perspective of the presentation of self (Grier and Campbell 2000), with its "trolls" and "flame wars" particularly helpful in revealing the tensions and controversies within the community (Coleman 2012).

The main topics of these tensions and controversies were the issue of software and the scientific modeling activity. From an epistemological point of view, an “epochal break” (Nordmann 2011) of scientific modeling activities, from a culture of explanation to a culture of prediction (Johnson and Lenhard 2011), has been linked to the availability of the ubiquitous desktop computer, thus empowering computational science practitioners with respect to expert computing scientists, equipped with supercomputing facilities. I hereby argue, following previous work on organizational software (Pollock and Williams 2008) that the scientific modeling software concomitantly turned into the

process of "generification" (Pollock et al. 2007), unveiling the mutual shaping of the modeling scientific activity and the technological device, thus provoking many tensions in the scientific computational community.

References

John D. Bolcer and Robert B. Hermann, 'The Development of Computational Chemistry in the United States', in Reviews in Computational Chemistry, ed. by Kenny B. Lipkowitz and Donald B. Boyd (John Wiley & Sons, Inc., 1994), V, 1–63.

Eric Francoeur, 'Cyrus Levinthal, the Kluge and the Origins of Interactive Molecular Graphics', Endeavour, 26 (2002), 127–131.

Donald B. Boyd, 'How Computational Chemistry Became Important in the Pharmaceutical Industry', in Reviews in Computational Chemistry, ed. by Kenny B. Lipkowitz and Thomas R. Cundari (John Wiley & Sons, Inc., 2007), XXIII, 401–451.

Jan K. Labanowski, 'Free Speech, Quality Control, and Flame Wars', Academe, January 2007.

D.A. Grier and M. Campbell, 'A Social History of Bitnet and Listserv, 1985-1991', IEEE Annals of the History of Computing, 22 (2000), 32–41.

Gabriella Coleman, 'Phreakers, Hackers, and Trolls: The Politics of Transgression and Spectacle.', in The social media reader, ed. by Michael Mandiberg (New York: New York University Press, 2012), pp. 99–119.

Alfred Nordmann, Science Transformed? Debating Claims of an Epochal Break (University of Pittsburgh Pre, 2011).

Ann Johnson and Johannes Lenhard, 'Toward a New Culture of Prediction: Computational Modeling in the Era of Desktop Computing.', in Science Transformed?: Debating Claims of an Epochal Break, ed. by Alfred Nordmann, Hans Radder, and Gregor Schiemann (University of Pittsburgh Pre, 2011), pp. 189–200.

Neil Pollock and Robin Williams, Software and Organisations: The Biography of the Enterprise-Wide System or How SAP Conquered the World, 1st edn (Routledge, 2008).

Neil Pollock, Robin Williams and Luciana D'Adderio, 'Global Software and Its Provenance: Generification Work in the Production of Organizational Software Packages', Social Studies of Science, 37 (2007), 254–280.

Verification & Validation of Computer Simulations: A Philosophical Analysis

Julie Jebeile (julie.jebeile@gmail.com)
IHPST, Paris, France

It is commonly admitted that the sanctioning of scientific theories or models is based on two distinct steps. The first step consists in testing whether the mathematical equations of models are correctly solved. It boils down to checking if the solutions we find to the equations are exact or almost exact. The second step consists in confirming or invalidating theories or models by verifying that the exact solutions to their equations fit with the experimental data on the natural or social systems under study. In general, philosophers focus on the second step, but the first one is also very important in the process of sanctioning scientific representations.

When these two steps are not performed distinctively, one after the other, we can face two typical problems. First, it can be difficult to assess the suitability of a model. We assess the suitability of a model by measuring the discrepancy between the calculated solutions and the experimental data. This measured discrepancy can integrate, at least partially, the discrepancy between the calculated solutions and the principled exact solutions. In such a situation, when a model provides us with wrong predictions, we do not know whether to blame the model, or the resolution of equations. Conversely, when a model provides us with good predictions, it might be because of calibrations introduced in the model to obtain better agreement with experimental data. Therefore a second problem emerges, that is the risk to take for correct numerical solutions which actually deviate from the empirical world because of their latent divergence from the model. Because of these two hypothetic problems, a clear separation of the two steps is required in the process of sanctioning scientific representations. Let us now see what it actually is in the sanctioning of simulation models.

How do engineers concretely sanction simulation models? One of their recent approaches, the Verification & Validation approach (V&V), has proven itself, notably by ruling the sanctioning of simulation models in the nuclear engineering sector (Oberkampf, Trucano, Hirsch, 2004). As its name suggests, V&V has two phases, i.e. the verification phase and the validation phase, which roughly correspond to the two steps of sanctioning representations described above. The verification phase aims to quantify the shift between the computer code and the theoretical model of which the code is the implementation. This shift corresponds to discrepancies between the ap-

proximate solutions provided by the computer code and the solutions that would have been ideally obtained if one had been able to perform the calculations exactly. As for the validation phase, it consists mainly in comparing a target set of numerical results, either directly with a database of experimental measurements, or with a set of results obtained with other codes which have already been validated. These latter are known as benchmarks and are useful to overcome the lack of experimental measurements. In this paper, I first show that the verification and the validation phases are hardly distinguishable in sanctioning simulation models. Furthermore, I contend that the entanglement between these two phases leads not only to the two already mentioned problems, but also to new problems specific to computer simulations that I shall present.

In the first part of the paper, I argue that the verification and validation phases are not two separable processes. This idea has already been emphasized by Winsberg (2010) but my argumentation here is different from his. For me, mainly, the entanglement of the two phases relies on the fact that, generally, in the verification phase, no precise assessment of discrepancies between the approximate solutions provided by computer and the principled exact solutions can be done. Concretely, the verification phase consists in proving the consistency, stability, and robustness of computer simulations which can not lead to any precise measurement of discrepancies. First, consistency is proven if the discretized equations approach the corresponding differential equations when space and time steps tend to zero and if the discretized equations have the same symmetry properties than the differential equations (Farge, 1986, p. 163). Discretizing the differential equations consists in turning them into approximate algebraic equations and is required in order to integrate them numerically.

Secondly, the stability of computer simulations is verified by checking that the simulation process does not amplify computer round-off, and further, does not risk diverging. Then, robustness is proven when the solutions of the computer simulations converge to the exact solutions of the partial differential equations. Proving consistency, stability and robustness of computer simulations is neither a rigorous way nor a precise way of quantifying the shift between the computer code and the model. The reason is, on the one hand, that all-

encompassing proofs of correctness, such as those developed in mathematical analysis and logic, do not exist in complex computer code (Oberkampf, Trucano, Hirsch, 2002). The sources of error and uncertainty in the computer simulations, related to the problems of consistency, stability, convergence, but also to the problems of existence, uniqueness of mathematical solutions,

and accuracy (computer round-off, insufficient discretization, truncation errors), can hardly be assessed formally (Farge, 1986). On the other hand, computer simulations are not open to direct inspection: they are epistemically opaque (Humphreys, 2004, p. 147). We cannot examine or justify every step of the computational processes that produce the outputs of computer simulations.

In the second part of the paper, I show that the entanglement of the verification and the validation phases not only makes more difficult to blame whether the model or the resolution is responsible for wrong predictions. But also this entanglement can favor the risk of being abused by wrong simulations for reasons that are specific to computer simulations. Take the example of discretization errors, defined as the differences between the exact solutions to the discretized equations and the exact solutions to the original partial differential equations. These errors are often poorly characterized or even ignored because they can hardly be assessed formally (Roy, 2010). However, the discretization errors can impact the physical behavior of the system observed on screen. Thus, for example, discretization errors in hydrodynamics models sometimes damp out the turbulent fluid motions. In such case, it can be very difficult to separate out modeling effects from discretization errors. This is generally the case when the physical model is connected with the choice of meshing, and especially the mesh size (e.g. large eddy simulation of turbulence).

In the third part of the paper, I show that the entanglement of the two phases brakes the "predictive turn" in computer-assisted sciences. It is commonly admitted that computer simulations extend our ability to investigate various natural and social systems in that they overcome our lack of empirical data. However, this claim ought to be weakened given the entanglement of the verification and validation phases. Indeed, the reliability of the results to simulations performed beyond the physical domain covered by the benchmarks is generally said to be warranted by theoretical extrapolations. But the accuracy of these extrapolations depends on the accuracy of the numerical scheme. It seems that, today more than yesterday, the reliability of new results drawn from our models narrowly depends on the benchmarks at our disposal, and thereby on our progress in empirically probing the world.

References

Farge, M. (1986) L'approche numérique en physique. Philosophia Scientiae, 7(2), 155–175.

Humphreys, P. (2004) Extending Ourselves. Computational Science, Empiricism, and Scientific Method. OUP.

Oberkampf, W. L., Trucano, T. G., Hirsch, C. (2004) Verification, validation and predictive capacity in computational engineering and physics. Applied Mechanics Review, Volume 57, Issue 5, 345.

Roy, C. (2010) Review of Discretization Error Estimators in Scientific Computing. 48th AIAA Aerospace Sciences Meeting, Orlando, FL, Jan. 4-7, 2010.

Winsberg, E. (2010) Science in the Age of Computer Simulation. University of Chicago Press.

Instruments of Control: Political Institutions and Information Technology at War

Jon Lindsay (jonrlindsay@gmail.com)
University of California San Diego, USA

Computer technology is everywhere in practical politics, but it receives comparatively less attention in political thought. Modern government and corporate bureaucracies are pervasive users and shapers of computing infrastructure, and they heavily depend on information technology (IT) to exert control over political behavior. There is a large and uneven popular literature on the impact of the information age on economic affairs, democracy, and warfare; this has fostered niche debates among scholars on the political impact of IT. Another family of research treats IT as a dependent rather than an explanatory variable in order to explain the effects of political economy on information systems and internet behavior. This work has deepened understanding of a critical technology, yet it tends to understate the intimate historical relationship between IT and political institutions.

This paper presents a participant-observer study of information work in a military special operations unit deployed in Iraq in 2007-8. By combining ethnographic methods with insights from the sociology of technology, I use this case to construct both applied and general political theory: in the first instance to understand the organizational implications of military dependence on IT; and in the second to articulate a more general theoretical account of how sociotechnical institutions can both enhance and degrade political control. In tracing the processes used by this organization to gather and analyze intelligence about insurgents in order to attack them, I describe the ways in which humans and machines jointly implement core institutional functions of measurement, coordination, and enforcement to achieve organizational goals. I then identify general conditions under which sociotechnical

institutions, which will be defined further herein, are more likely to improve or undermine control.

Military organizations and ideas about their effectiveness are important to study in their own right because of the significant political and economic costs of waging and preparing for war. They are also, furthermore, useful sites for exploring more general political and organizational phenomena. In this case, the ability to perform effectively on the battlefield is one extreme–and extremely coercive–type of political control. In the 1990s and early 2000s there was vigorous debate in American defense policy circles about whether advances in information technology (IT) created a revolution in military affairs (RMA). Proponents argued that IT required militaries to undergo an ambitious program of "defense transformation" in order to implement a potent new doctrine of network centric warfare (NCW). These technological visions have found renewed expression in contemporary debate over the role and effectiveness of remotely piloted weapons (drones) and the risk of cyberwarfare to internet-dependent societies. A common theme across these is that IT improves military effectiveness by enhancing perception of enemy vulnerabilities and the ability to act from a distance quickly upon them to achieve very specific effects. In the classic RMA vision this is accomplished through networks of battlefield reconnaissance sensors and long-range precision-strike weapon systems. Armed drones with long loiter times and high acuity–and without humans aboard–are a natural elaboration of the classic vision. Cyber weapons are now thought to offer, through the global reach and readily available tools on the internet, an even more radical form of long-range reconnaissance and precision attack. These related ideas about the efficacy of IT in future war continue to foster debate within the security studies field.

Another commonality across these new forms of warfare has attracted far less attention by contrast. For network-centric forces, drone operators, and cyber warriors alike, warfare is increasingly experienced remotely through the mediation of digital data rather than through bodily presence on the battlefield. Personnel spend an increasing amount of their time, and an increasing proportion of the military labor force works exclusively, in bureaucratic settings located at some distance from combat action. Their work within offices and in front of computer screens can, nevertheless, still exert tangible effects on combat action, but this action is indirect through organizational distribution and technological mediation. One important question is whether this increasing immersion in technology enables personnel to perceive the environment more or less accurately and prosecute their missions more or less effectively.

Participant-observation offers a methodology to observe such phenomena in situ. While deployed in Iraq with a special operations unit for seven months, I was able to observe its measurement, coordination, and enforcement mechanisms in action, implemented with both human and technological means. In contrast to the conventional belief that technology can "lift the fog of war," I found that the unit used its information systems to construct and act upon a world consistent with its deeply ingrained heroic commando identity. The "unblinking eye" of high-tech intelligence, surveillance, and reconnaissance (ISR) did not just reveal an objective "battlespace" but rather reflected and reinforced prior institutional preferences for killing "bad guys." Coupled with relative organizational autonomy for picking and prosecuting targets, this situation promoted behavior at odds with ostensible national counterinsurgency objectives focused on "winning the hearts and minds" of the indigenous population.

Close attention to the ways in which this one particular community used IT to exert control, in the context of sociological and historical studies of computer users elsewhere, suggests that more information about the environment and more computational processing power may not necessarily improve understanding of the world. In contrast to popular expectations about improved transparency and effectiveness through IT, the organizational context in which IT is embedded strongly biases the way it is used. Institutionalized goals and practices shape the types of problems technology is used to solve and a community's capacity to use technology to solve them. In cases where an organization can agree internally about how to use technology and when that use is congruent with real constraints in the environment (i.e., when people agree on the solution to a problem and that problem is solvable in technical principle), then performance can be enhanced. If, however, organizations suffer from some pathology, such as a mismatch of warfighting preferences with the demands of the warfighting problem, then technology will simply amplify that pathology.

Philology of Programming Languages

Baptiste Mélès (baptiste.meles@normalesup.org)
Archives Henri-Poincaré, Université de Lorraine, France

Programming languages are often considered as mere formal languages, i.e. as languages which are only defined by a set of rules of formation and transformation. This is why abstract tools such as Turing machines, lambda calculus and their variations can be – very fruitfully – used to modelize

them. This modelization is a simplification, which deliberately drops some features, considered as unessential to programming languages as such. But are these "unessential" properties so unessential ?

"Real-life" programming languages such as assembly, C, Perl and Java indeed have many characteristics which nobody would want in a honest formal language :

1. Their syntax has many irregularities. For instance, in C language, all functions must have an explicit type – except void functions, which, for historical reasons, do not need to be declared as such. In a formal language, no exception like that should be accepted without a loss of clarity.

2. Their syntax is often widely redundant : they often have as well for as while loops, and many signs of "syntactical sugar." Perl even has an until loop, which is exactly the same as a while followed by a negation. In a formal language, the principle of economy should force us to choose between equivalent structures ; otherwise, the language would loose every conceptual purity.

3. They even have historical residues : C-like syntax is still used in many younger languages, such as C++, Java, Perl and JavaScript ; the obsolete register keyword still belongs to the C language. Formal languages are expected to derive only from conceptual considerations, and should not be weighed down by historical contingences.

4. Their abstract and complete formal definition (as given in Backus-Naur normal Form) usually comes long after their first definition and their current use. ANSI C and XHTML came long after C was used to program operating systems, and HTML to encode web pages.

5. They are not learned through their abstract definition, but with "Hello world"s, examples and practice. Even worse : most of their users – those who do not write compilers – do not even care about their formal definition.

Real-life programming languages are thus quite far from their abstract modelization. This fact should draw our attention.

Are these properties really so uninteresting ? They unexpectedly make these languages similar to natural languages, such as Russian or Spanish. Natural languages indeed have many syntactical irregularities ; they have syntactical

redundancies ; they are usually learned without any complete formal definition, just with practice and examples. And they are full of his- tory : one must learn Latin to understand where Italian and French struc- tures come from, and know rudiments of Ancient Chinese to understand where Chinese characters and words come from.

Moreover, nobody complains about these features of natural languages : they make languages interesting, and even give way to style and idioms. Men are not bees : they can choose between several ways of expressing the same meaning.

Programmers are not bees either. They can choose between several ways of writing the "same" program. Programming have styles. Programmers even have grammarians which warn them against the use of programming structures such as goto. A program must not only be efficient : it must be elegant.

We will thus try to show some concrete examples of linguistic and phi- lological concepts which can be fruitfully applied to the concrete study of programming languages.

This will lead us to reverse Chomsky's perspective : while he probably thought he honoured English by describing it as a formal language, we intend to show that C++ and Perl, far from being failed formal languages, possess, in their structure as well as in their history, some of the most beautiful and exciting properties of natural languages.

Tomás Maldonado and the Sign System for Olivetti ELEA 9003

Elisabetta Mori (bettygorf@gmail.com)
Università degli Studi di Firenze, Italy

In this contribution we present a paradigmatic case study in the relation between Computer Science and Design History. The overall aim is to prove that the integration of different areas may result in a more comprehensive approach to the development of early computing machines. ELEA 9003 is the acronym for Elaboratore Elettronico Automatico. Together with Calcolatrice Elettronica Pisana (CEP) - presented to the public later, in 1961 - it was one of the earliest Italian Computers, produced serially by Olivetti beginning in 1959 (Cignoni 2012). The interest in this mainframe computer spans several fields, as its history lies at the intersection of the birth of Italian information science, the ergonomic design of computing machines, and Adriano Olivetti's ideals and philosophy about industry and society (Mori

2013).

The machine, one of the first transistorized mainframe computers in the world, was built by a team of engineers and physicists led by Mario Tchou, a Chinese electronic engineer, born in Italy but trained in the United States (Rao, 2005). The architect Ettore Sottsass Jr., in collaboration with the Dutch industrial designer Andries Van Onck, was in charge of the aesthetic and ergonomic design of the machine (Sottsass 1983; Van Onck 2005). The architect reversed the relationship between man and machine, putting the user instead of the computer at the center of the project, resulting in an innovative design. Fragments of this story have already been told: ELEA 9003 was a paradigm of excellence in Italian research (Soria 1979; Rao 2005; Filippazzi 2006). What is still mostly unknown is that Tomás Maldonado, in 1960, developed a symbols system for the console of Olivetti ELEA 9003 (Anceschi 2009).

The console was made of a keyboard and a display: the indicators were identified by Italian words. Ambitiously, Olivetti aimed to launch its brand new mainframe computer in the international market together with IBM, Ferranti, Siemens, Bull, and the like. Instead of translating Italian abbreviations into English, Olivetti thought of a brand new 'international' solution. In order to export this machine to foreign countries, Olivetti asked Maldonado to elaborate a sign system, which could be easily learned by any operator, regardless of his mother tongue – a novel language to be used to communicate between man and machine (Riccini 2010).

At the time Tomás Maldonado was the director of the Ulm School of Design (Hochschule f ür Gestaltung Ulm), Germany, one of the most progressive educational institutions of design in the Fifties and Sixties and a pioneer in the study of semiotics. Together with Gui Bonsiepe, Maldonado designed a system of logograms, corresponding to nouns, verbs, and adjectives. He chose logograms because they are non-spoken characters, surpassing any national language. They were designed so that they could be learned by means of a language, but at the same time they were not tied to any particular one (Bonsiepe 1961; Krampen 1965). The project was interesting but complex and ambitious: the sign system was designed with more than a hundred logograms but in the end they had never been applied to ELEA 9003 (Riccini, 2010).

The Olivetti Electronic Department (Divisione Elettronica Olivetti - DEO) eventually lost its leaders and supporters in the company: both Adriano Olivetti and Mario Tchou died suddenly in 1960 and 1961, respectively. Due to the lack of leadership in the company - together with bad administration,

several political issues, and the wrong evaluation of the future business of computers - the Olivetti Electronic Department was entirely sold to General Electric in 1964, stopping Italian research in the field (Soria 1979). All this effort put into the elaboration of a sign system for the interaction of man and machine - with both a grammar and a syntax, through a visual code - vanished upon the quick development of electronics and computers with ever simplified man-machine interfaces. It was, however, Olivetti who understood the importance of a logogrammatic communication system - independent of speech words - applied to computers, several years in advance, as stated by Maldonado in a recent interview (Riccini 2010).

Our aim will be to present the importance of design and semiotics to the early development of computing machines in Italy, through rare or unknown facts, as well as graphics and unpublished original photos, all from various archives, systematically ordered for the first time.

References

Anceschi, G. (2009). Maldonado semiotico della conoscenza. E|C Serie Speciale, III - 3/4, 207-214. www.ec-aiss.it

Bonsiepe, G., Maldonado, T. (1961). Sign System Design for Operative Communication. Uppercase, 5, 11-18.

Cignoni G.A., Gadducci F. (2012). Rediscovering the Very First Italian Digital Computer. In 3rd IEEE History of Electro Technology Conference, IEEE.

Filippazzi, F. (2006). Quel computer nato tra i cavalli. In L. Dadda (ed.) La nascita dell'informatica in Italia, Milano: Polipress.

Krampen, M. (1965). Signs and Symbols in Graphic Communication. Design Quarterly, 62, 1-31.

Mori, E. (2013). Ettore Sottsass Jr. e il design dei primi computer Olivetti. AIS/Design Storia e ricerche, 1. www.aisdesign.org

Rao, G. (2005). Mario Tchou e l'Olivetti Elea 9003. PRISTEM Storia, 12-13, 85-119.

Riccini, R. (2010). Un'impresa aperta al mondo. Conversazione con Tomás Maldonado. In G. Bigatti & C. Vinti (ed.) Comunicare l'impresa. Cultura e strategie dell'immagine nell'industria italiana (1945-1970). (134-152). Milano: Guerini e Associati.

Soria, L. (1979). Informatica: un'occasione perduta. La divisione elettronica dell'Olivetti nei primi anni del centrosinistra. Torino: Einaudi.

Sottsass, E. (1960). Forme nuove per i calcolatori elettronici. Notizie Olivetti, 68, 27- 29.

Sottsass, E. (1983). Storie e progetti di un designer italiano. Quattro lezioni di Ettore

Sottsass Jr. A. Martorana (ed.). Firenze: Alinea Editrice. Tomás Maldonado (2009). Catalogo della mostra. Milano: Skira.

Van Onck, A., Takeda, H. (2005). Avventure e disavventure di design. Firenze: Alinea Editrice.

Cybernetics, control and Big data

Teresa Numerico (teris@mclink.it)
University of Rome 3, Italy

The advent of "Big data" (Mayer-Schonberger, Cukier 2013) raises new questions about the cyber-utopia of a brave new open cyberspace. In this talk I propose a genealogy of the network starting from cybernetics: the idea of concentrating on the special case of communication represented by control– suggested by Norbert Wiener in his seminal book of 1948 – left only a little hope that cyberspace could allow a special freedom experience, with respect to real life.

Wiener played an ambivalent role in the shaping of communication technologies: he allowed the idea of obtaining control tools by developing communication devices, while suggesting that these machines could be very dangerous because they can favor concentration of power, and mechanization of human behavior. Cybernetics' crucial point suggested that there was no boundary and no relevant difference between a biological organism and a mechanical device as far as they shared a similar structure to interact with the environment. The interaction structure, common to both fields of research was negative feedback, and the black box assumption which included the orientation toward a purpose (teleological behavior) and the capability of being affected by the external context by transforming the agent behaviors to "adjust future conduct" to the requests of the environment.

As acknowledged by the famous paper Behavior, purpose and teleology: "The methods of study for the two groups [living organisms and machines] are at present similar. Whether they should always be the same may depend on whether or not there are one or more qualitatively distinct, unique characteristics present in one group and absent in the other. Such qualitative differences have not appeared so far" (Rosenblueth, Wiener, Bigelow 1943, p. 22). If there is no difference between machines and human beings and if we limit communication to the special case of control, a machine and a human

being who is in charge to obey orders (in the military or in other similar contexts) are perfectly equivalent interlocutors. Wiener was also concentrated on the struggle against the secrecy policy on scientific knowledge and on the excess of patent law in protecting the wrong actors of the innovation process. According to him "information is more a matter of process than of storage" (Wiener 1950/1954, p. 121). His approach to management and organization of knowledge was guided by the awareness that scientific discovery depended on the availability of the information on which new achievements were based. That is why he was strongly in favor of what would be called today 'open science'.

However things have changed radically from the times he was writing, communication technologies now include the ability of preserving data as dynamic entities, as in Big data management. There is now no opposition between storage and process and we can pretend to store the process in its dynamics. Moreover if it is true that the collectivity would not benefit from the emergence of the 'enclosures' of knowledge productions, there are a lot of business agents who will exploit the intellectual property regulations.

Wiener was rather worried about the consequences of cybernetics approach that allowed the breach of boundaries between human beings and machines, as he was well aware of the possible use of communication technologies as tools for the concentration of power. While in 1960, J.C.R. Licklider introduced the concept of "man-computer symbiosis" under the influence of the cybernetic notions of communication and feedback, which was the model for the interaction between the machine and its user. In a famous letter to the 'intergalactic network', written in 1963, he connected military command and control techniques with the requests of scientists for the inclusion of the computer in the formulative thinking process: computers would help scientists, as new colleagues, in suggesting scientific models to make sense of data, not only in dealing with calculations. In order to achieve this goal he thought that it was necessary to define a language which was at the same time easy to understand for human beings and adequate to interact with the machine. The model of this language was anticipated by Leibniz early theory: the creation of a 'Calculus Ratiocinator' that could calculate all the proofs needed using the language of a 'Lingua Characteristica', in which each notion possessed an unambiguous label. This mirage, perfectly embodied by the computer as a Laplacian machine, seemed temporary defeated by the network as an interactive tool, whose protocols were open, transparent, and impossible to govern by a single authority. However, as suggested by Wiener, the idea of communicating with or by a machine, while the machine can substitute human beings in terms of production of meanings, creation

of discourse and storage of propositions about behavior habits risks to increase control over the network. Interacting with and by the machine implies inevitably both controlling and being controlled by the interface device.

Big data represent the realization of Licklider's dream and Wiener's nightmare: obtaining patterns of correlations between data without a theory and only by algorithmic analysis of the quantitatively enormous amount of information. The availability of such a big amount of data on people social habits is made possible by the advent of Content Management System (CMS), which was one of the crucial technologies of the so-called Web 2.0. It represented the key feature for organizing all the users' free content in the rigidity of a database, whose data set now constitutes the gist of the Big data "revolution". This structured organization of information reproduced the Laplacian machine within the network, with the aid of cloud computing.

References

Licklider, J.C.R. (1960): "Man-computer symbiosis" in IEEE Transactions on human factors in Electronics, Vol. HFE-I, March 4–11. http://memex.org/licklider.pdf.

Licklider J.C.R. (1963) Memorandum for members of the affiliated of the Intergalactic Computer Network. http://packet.cc/files/memo.html.

Mayer-Schonberger V., Cukier K. (2013) Big Data: A Revolution That Will Transform How We Live, Work, and Think, Eamon Dolan/Houghton Mifflin Harcourt, Boston.

Rosenblueth A., Wiener N., Bigelow J. (1943) "Behavior, Purpose and Teleology", in Philosophy of science, Vol. 10, pp. 18-24.

Wiener, N. (1948/1961): Cybernetics: or Control and Communication in the Animal and the Machine. MIT Press, Cambridge (Mass).

Wiener, N. (1950/1954) The human use of human beings, Houghton Mifflin, Boston.

What network computing does to communication
A retrospective analysis of early debates confronting and inventing online communication ethics

Camille Paloque-Berges (camille.paloque_berges@cnam.fr)
CNAM, Paris, France
Haud Guegen (haud.gueguen@cnam.fr)
CNAM, laboratoire DICEN, Paris, France
Claire Scopsi (claire.scopsi@cnam.fr)
CNAM, laboratoire DICEN, Paris, France

In the field of network and digital communication technologies, the fast pace of innovation should be questioned in critical terms about how they change rules and practices of communication. The meeting of the social and the technical in Internet communication has been interrogated in various disciplines such as sociology and information sciences. We would like to ask how the technical meets the ethical in online communication. What is the relationship between the instrumentality of computer-mediated communication and the norms of network interaction and exchange ? How does one behave online when communication does not rely only on verbal codes but also on the formal techniques of computer languages and the double bind of network computing and interfaces ?

We take up the discussion suggested by James H. Moor in his essay "What is Computer Ethics ?" so as to ask how network computing can change the nature and the rules of human communication. We confront the relevance of two theories about the ethics of communication, discourse ethics (Habermas 1991) and recognition theory (Honneth 1996), in the context of computer-mediated communication for collective, forum-type discussions. Thus, we would like to contribute from a transdisciplinary standpoint, at the frontier between philosophy and communication sciences, to these social and human sciences debate : what does the Internet do to communication ? What kid of rules and politics emerge from these new forms of communication ? To what extent these rules are different from face-to-face communication, and how can they be interpreted both in continuity and alteration ?

An retrospective look on digital networked communication We think it would be useful to go back to studying online exchanges in electronic forums in the first half of the 1990's in order to shed a light on the present. We chose to analyze French speaking newsgroups on Usenet, which

were born in 1993 in a crucial transition period named "Eternal September" in Internet folklore, that is the moment when an afflux of new users came to know and participate in online discussions whilst the Internet is discovered through new protocols, software and interfaces brought about by Web technologies. As the Internet becomes more popular, the relationship between the network medium and its use becomes more sensitive and noticeable as new converts look for answers to solving problems in network communication uses and confront their own frames of references and norms to those developped by experienced users way before the Web came to be. These real-time interactions happen at a moment when network communication is far from being stabilized, and undergo a transitional process : digital networks as a communication medium is being reinvented to accomodate both new uses and new techniques.

We will question this interaction by studying socio-technical mediations in non-web forums under the predicate that the architecture of forum-type communication systems, even if instable, carry normative social and relational models (Voirol 2010). This retrospective on a historical standpoint in Internet communication seems relevant in order to study closely the "malleability" associated with computing ethics according to Moor.

Forums as witnesses to online discursive exchanges We analyze a set of Usenet newsgroups, whom users used to discuss Internet and network computing themes from 1993 (French newsgroup on Usenet opened this year) to 1995 (year when the 95/46/CE directive on personal data use was adopted by the European Parliament, with a notable reference to network electronic telecommunications). We attend to show how moral issues appear while collective exchanges are being regulated (these issues are related to the way a person, considered a network user, is connected to a group in a set of good practices of communication). On the other hand we show how these issues are linked to ethical problems needing to be ruled on an upper level and formalized by laws, and social norms (censorship vs free expression for example). This will lead us to also analyze the negative side of these moral issues : how communication is contaminated by pure instrumental actions (Habermas) and phenomena as disrespect and denial of recognition (Honneth, Voirol, Granjon). Examples of case-studies will range to conflicts about how to present oneself or one's ideas in the Usenet newsgroups by instrumenting network communication to how a group of users exchange technical tips to block unwanted users or information.

We will first perform a discourse analysis on the interlocutors' statements,

including specifically a semantic study of the lexical fields of ethics and an analysis of their evolution in time. In order to do so, we use qualitative and quantitative methods adapted for network forum studies (including a software tool called Calico). We will compare our results with grey literature's contents devoted to the regulation of the "netiquette" (charters, "FAQ") in groups. We supplement the analysis of themes related to ethics with a close study of the affordances and appropriation of forum tools by users from the point of view of the semiotics of technical communication. Our focus will be the "device of enunciation" (Jeanneret, Souchier, 2005), by listing in particular the technical operations proposed or performed by the interlocutors in order to put into practice their understanding of communication rules. We will be aiming especially at the way users technically operate in order not only to deal with the ergonomic of the mailing system, and the virtual communication but also with the hardware and software capabilities in the network (servers, storage..).

From the premise that some ethical issues in communication are due to network-mediated computing, we state that network-mediated communication frameworks, just as their social and technical regulation is called into question, make the promise of a discursive space driven by a communicative action (which thus allows to negotiate rules). However, in these frameworks, actual relationships are also seemingly driven by "instrumental rationality" which constantly redefine the shape of the moral rules being assigned to the communication behavior of the group, and thus permeates the negociations driven by "communicative action (or rationality)" (Habermas 1983). In order to solve this dichotomy and perceive the ethical malleability of these frameworks, we will try to present the idea of "technical argument" as a speech act (in the Austin sense), performative in a symbolic sense as much as in an operational sense, which takes place in both technical gestures and discursive action performed in network communication. As an outcome, we will ask if the ethical questions linked to network-mediated communication devices in the middle of the 90's are discourses accompanying innovation in order to enhance an ideological imaginary of Internet (Flichy 2007) or if they are more widely the expression of a revolution in communication.

At last, we will put into perspective the analysis produced during this historical period of the growth of Internet with ethical reflections about network-mediated communication applied to contemporary social media. Current writing reagrding the ethics of recognition will lead us to ask whether network computing gives a push to mutual recognition amongst social actors or, on the contrary, helps new forms of disrespect and domination to be expressed (Granjon 2012).

References

Patrice Flichy (2007). The Internet Imaginaire. Translated by Liz Carey-Libbrecht. Cambridge :MIT Press

Fabien Granjon (2012). Reconnaissance et usages d'Internet. Une sociologie critique des pratiques de l'informatique connectée, Paris, Presses des Mines, 2012.

Jürgen Habermas (1991). Moral Consciousness and Communicative Action. Cambridge: MIT Press

Jürgen Habermas (1984). Theory of Communicative Action, trans. Thomas McCarthy, Boston: Beacon Press

Axel Honneth (1996).The Struggle for Recognition: The Moral Grammar of Social Conflicts. Polity Press

Jeanneret Yves et Souchier Emmanuël (2005). "L'énonciation éditoriale dans les écrits d'écran", in Communication et langages. No145, 3ème trimestre, pp. 3–15.

Olivier Voirol (2010). "Digitales Selbst: Anerkennung une Entfremdung", WestEnd : Neue Zeitschrift für Sozialforschung, pp. 106-120.

Computer Science between Science and Technology: A Red Herring?

Marcello Pelillo (pelillo@dais.unive.it)
Ca'Foscari University of Venice, Italy
Teresa Scantamburlo (scantamburlo@dais.unive.it)
Ca'Foscari University of Venice, Italy
Viola Schiaffonati (schiaffo@elet.polimi.it)
Politecnico di Milano, Italy

Computer science has been plagued since its beginnings by the elusiveness of its very nature, being halfway, as the name itself implies, between science and technology. Dijkstra, for example, insisted on de-emphasizing the role of the machine stressing the intrinsic abstract character of the field; others held that the 'science' in computer science is a misnaming, given its engineering nature. The debate still goes on but, in time, the interdisciplinary nature of computer science has been widely recognized and, accordingly, it is now defined partly as scientific, partly as mathematical, and partly as technological (Denning 2005). There are some subfields, however, in which the mutual exclusiveness of the scientific and technological paradigm is still dominant. This is quite evident in some areas of artificial intelligence, such as machine learning and pattern recognition, where only few systematic attempts to understand the

interplay between technological and scientific factors have been made. In this paper, we attempt to approach the question by making use of some recent developments in the philosophy of technology and in the philosophy of science. Our analysis will be complemented by historical examples taken from the field of artificial intelligence.

Pattern recognition and machine learning face a broad spectrum of problems involving the ability to discover regularities in data, generalizing from observations. In these two areas many traits of the traditional opposition between science and engineering are still present. Although some scholars pointed out both scientific and technological aspects (Duin et al. 2007), the most common tendency is to emphasize a single component. In some cases, especially in the past, the approach of pattern recognition and machine learning has been associated to the scientific practice of physics (Serra 2000) or more generally of experimental sciences (Langley 1988). On the contrary, nowadays, it prevails the idea that machine learning and pattern recognition are primarily engineering disciplines dealing with problems intrinsically dependent on the application they are built for (Duda 2001).

This sharp opposition between science and technology stems from an oversimplified view of their mutual relationship. In the light of some new achievements in the philosophy of technology (Franssen et al. 2010), it turns out that, granted that there are indeed important differences, at the conceptual level the boundary between the two camps is more blurred than is commonly thought, and that they stand to each other in a kind of circular, symbiotic relationship. Technology can be considered an activity producing new knowledge on a par with ordinary science (Simon 1969). The so called operative theories (Bunge 1966) in technology look like those of science and their contribution goes beyond the mere application of scientific knowledge. Conversely, even science can be brought closer to technology when its progress is expressed in terms of immanent achievements. This idea lies at the heart of the problem solving approach (Laudan 1977) and could well characterize much of the work in the fields of machine learning and pattern recognition.

Our discussion will advocate that both machine learning and pattern recognition are suitable examples of the circularity joining scientific and technological efforts. If we look at the history of the fields, we observe that most the technological progress springs from very scientific issues and early attempts tried not only to provide feasible solutions, but also to uncover the structure of the problems. The case of neural networks is paradigmatic, as their formulation was been clearly inspired by scientific purposes, that is, by the wish of studying and imitating the brain but, in the phase of their

renaissance, technical matters prevailed. Indeed, with the (re)invention of the back- propagation algorithm for multi-layer neural networks and, above all, thanks to the impressive results obtained by these new models on practical problems such as zip code recognition and speech synthesis a new wave of excitement spread across the artificial intelligence community. At that point, however, it was widely accepted that these models had no pretention of being biologically plausible except of being interesting computational devices (Pavlidis 2003). Bayesianism is another interesting example of the gate allowing machine learning and pattern recognition to move from theoretical issues to more practical aims. Introduced as a theory, which can characterize the strength of an agent's belief, it provided many inference algorithms with a practical machinery. On the other hand, recent advances in density estimation techniques, such as nonparametric Bayesian methods, have been successfully applied to approach a variety of cognitive processes (Sanborn et al. 2010). This choice is typically useful in problems suffering from a combinatoric explosion and particularly suitable to bridge the gap between the computational and the algorithmic levels of rational models of cognition.

In conclusion, with the contribution of philosophy of technology and philosophy of science, we shall argue that we should rethink the classical dichotomy between science and technology, which is still holding in some subfields of computer science, as they appear closer than we used to think. Historical examples from artificial intelligence will suggest that computer science works as a bridge between the two, indeed, and many ideas from science result in technological innovation via computer science, and vice versa.

References

Bunge, M. Technology as applied science. Technology and Culture 7:329–347, (1966).

Denning, P. Is computer science science? Communications of the ACM 48(4):27–31, (2005).

Duda, R.O., Hart, P.E., Stork, D.G. Pattern Classification. Wiley, (2001).

Duin, R.P.W., Pekalska, E., The science of pattern recognition. Achievements and perspectives. Studies in Computational Intelligence (SCI) 63: 221–259, (2007).

Franssen, M., Lokhorst, G.-J., van de Poel, I. Philosophy of technology. In: Zalta, E. (Ed.), The Stanford Encyclopedia of Philosophy, (2010).

Langley, P. Machine learning as an experimental science. Machine Learning 3:5–8, (1988).

Laudan, L. Progress and its Problems. University of California Press, (1977).

Pavlidis, T. 36 years on the pattern recognition. Pattern Recognition Letters 24: 1–7, (2003).

Serra, J. Is pattern recognition a physical science? Proc. ICPR'2000 3: 29–36, (2000).

Sanborn, A.N., Griffiths, T.L., Navarro, D.J. Rational approximations to rational models: alternative algorithms for category learning. Psychological Review 117(4):1144–1167, (2010).

Simon, H. The Sciences of the Artificial. MIT Press, (1969).

Von Mises, Church, and the Birth of Algorithmic Randomness

Christopher Porter (christopher.p.porter@gmail.com)
LIAFA - Université Paris Diderot – Paris 7, France

In 1919, Richard von Mises published in his 'Grundlagen der Wahrscheinlichkeitsrechnung' (von Mises 1919) a definition of randomness for infinite sequences that he in- tended to serve as a foundation for his theory of probability (which is more fully expounded upon in (von Mises 1964 and 1981). This definition was widely rejected as inade- quate by von Mises' contemporaries, who objected that it made use of an ill-defined notion of place selection (see, for instance, Kamke 1932, Kamke 1933, and Fréchet 1938). According to these objectors, von Mises' definition was highly arbitrary, as he never precisely specified what should count as a place selection, and this, they claimed, led to his definition being inconsistent.

What Von Mises' critics (as well as later commentators on his definition) failed to recognize was that this apparently arbitrary character of place selections was held to be necessary by von Mises. In von Mises' view, if one were to define random sequences in terms of a fixed, well-defined collection of place selections, the resulting theory of probability would be incomplete, incapable of solving certain problems in the probability calculus, thus failing to attain what one might call the resolutory ideal of completeness for theories of probability.

In attempting to respond to von Mises' critics, Alonzo Church suggested in his 1940 article, 'On the Concept of a Random Sequence' (Church 1940), that random sequences should be defined in terms of effectively calculable and thus (by the Church-Turing thesis) computable place selections, thereby providing the first definition of algorith- mic randomness. However, as this restriction of the collection of place selections to the computable ones is contrary to von Mises' prohibition against defining random- ness in terms of a fixed collection of place selections, the question arises as to whether Church was aware of von Mises' intention for his definition of randomness, namely to yield a theory of probability attaining the resolutory ideal of completeness.

The primary goal of this talk is to present an answer to the question as to how Church viewed his definition of algorithmic randomness vis-à-vis von Mises' original intention for his definition of randomness. The answer I suggest is uncovered in the brief correspondence in the early 1960s between Church and Hilda Geiringer (Church 1966a and 1966b, Geiringer 1966), von Mises' wife, herself a mathematician who edited a number of von Mises' works after his

death in 1953. As I will highlight, not only did Church recognize the centrality of von Mises' resolutory ideal of completeness to his larger programme, but he also subscribed to an alternative formulation of the resolutory ideal. According to Church, the problems of the probability calculus that occur in actual practice are those that can be solved by computable place selections. Thus, in his view, a limited version of the resolutory ideal can be attained simply by ignoring those problems of the probability calculus that are not solvable by algorithmic means, for in Church's view, these are not problems that we should worry about solving to begin with.

Church's application of computability theory to the study of randomness is thus not merely significant for technical reasons, but it also raises a number of more general philosophical questions about the role of computability in the solution of problems (and not just problems of the probability calculus): Should we restrict our attention to problems that can be solved effectively? Why are problems that are effectively solvable privileged over problems that are not? Is there anything lost by ignoring problems that cannot be solved effectively?

References

Alonzo Church. On the concept of a random sequence. Bull. Amer. Math. Soc., 46:130–135, 1940.

Alonzo Church. Letter to Hilda Geiringer von Mises, 11 june 1966. Alonzo Church Papers, Box 22, Folder 21; Department of Rare Books and Special Collections, Princeton University Library, June 1966.

Alonzo Church. Letter to Hilda Geiringer von Mises, 16 mar. 1966. Alonzo Church Papers, Box 22, Folder 21; Department of Rare Books and Special Collections, Princeton University Library, March 1966.

Maurice Fréchet. Exposé et discussion de quelques recherches récentes sur les fondéments du calcul des probabilités. In Colloque consacré au calcul des probabilités, volume 735 of Actualités Scientifiques et Industrielles, pages 22–55. Hermann, 1938.

Hilda Geiringer. Letter to Alonzo Church, 13 apr. 1966. Alonzo Church Papers, Box 22, Folder 21; Department of Rare Books and Special Collections, Princeton University Library, April 1966.

Erich Kamke. Einführung in die Wahrscheinlichkeitstheorie. S. Hirzel, Leipzig, 1932.

Erich Kamke. Uber neuere begründungen der wahrscheinlichkeitsrechnung. Jahresbericht Deutsche Mathematiker Vereinigung, 42:14, 1933.

Richard von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. Math. Z., 5(1-2):52–99, 1919.

Richard von Mises. Mathematical theory of probability and statistics. Edited and Complemented by Hilda Geiringer. Academic Press, New York, 1964.

Richard von Mises. Probability, statistics and truth. Dover Publications Inc., New York, English edition, 1981.

Exploring Thue's 1914 paper on the transformation of strings according to given rules

James Power (jpower@cs.nuim.ie)
National University of Ireland, Maynooth, Ireland

Rarely has any paper in the history of computing been given such a prestigious introduction as that given to Axel Thue's paper by Emil Post in 1947 (Post 1947):

> "Alonzo Church suggested to the writer that a certain problem of Thue (Thue 1914) might be proved unsolvable ..."

However, only the first two pages of Thue's paper are directly relevant to Post's proof, and, in this abstract, I hope to shed some light on the remaining part, and to advocate its relevance for the history of computing.

Thue Systems Thue's 1914 paper is the last of four he published that directly relate to the theory of words and languages (Berstel 1995, Stein and Thomas 2000). In this 1914 paper, Thue introduces a system consisting of pairs of corresponding strings over a fixed alphabet:

$$A_1, A_2, A_3, \ldots, A_n$$
$$B_1, B_2, B_3, \ldots, B_n$$

and poses the problem: given two arbitrary strings P and Q, can we get one from the other by replacing some substring A_i or B_i by its corresponding string? Post called these systems of "Thue type" and proved this problem to be recursively unsolvable.

Reception of Thue's Work Thue's earlier work was not widely cited but often rediscovered independently (Hedlund 1967), and something similar seems to have happened with the 1914 paper.

For example, Thue is not among the 547 authors in Church's 1936 Bibliography of Symbolic Logic, nor is Thue cited in Post's major work on tag

systems, correspondence systems, or normal systems before 1947. His work appears to have had no direct influence on the development of formal grammars by Chom- sky in the 1950s. Most subsequent references to Thue's paper (where they exist) note it only for providing a definition of Thue systems. Thue's awareness Thue explicitly understood the general meta-mathematical context (that we now associate with (Hilbert's programme)), describing the prob- lem as being of relevance to one of the "most fundamental problems that can be posed".

Further, he phrases the problem in terms that have become quite familiar in the post-1936 world:

> "... to find a method, where one can always calculate in a predictable number of operations, ..."

This language parallels that used in Hilbert's 10th problem in 1900 and places Thue's work firmly in what we would now regard as computing, rather than pure algebra.

Foundations of Language Theory Having posed the general problem in §II of his paper, Thue then presents an early example of a proof of (what we would now call) termination and local confluence for a system where the rules are non-overlapping and non-increasing in size.

When reducing some string P , we must find some occurrence of A_i and replace it with B_i . A difficulty arises if there is an overlap: some substring CUD in P , such that A_i matches both CU and UD, and thus choosing one option will eliminate our ability to later choose the other.

In §IV, Thue presents the string U as a common divisor of CU and UD and then shows how we can apply Euclid's algorithm to derive a Thue system from this. Euclid's algorithm had been considerably generalised throughout the 19th century, but here the string U "measures" the strings CU and UD just as Euclid's lines measure each other (Elements, Book 10, proposition 3).

Thue derives another *algorithm* in §V which, given two strings P and Q will derive those strings equivalent to them, and gradually reduce them to a core set of irreducible strings, providing a solution to the word problem in a restricted case. He investigates variants of these presentations based on their syntactic properties in §VI and gives some examples in §VII.

We remark that from the identity $CU \equiv UD$ we can derive rules of the form $CU \rightarrow UD$, and that this template is precisely what Post termed normal

form for his rewriting systems.

Thue's "completion" algorithm In §VIII of his paper Thue develops an algorithm to derive a system of equations from any given sequence R. This is interesting not just for its structure (the algorithm iterates until it reaches a fixed point) but also for its use of overlapping sequences as a generation mechanism.

Starting from some given identity sequence R we can identify all pairs where $R \equiv CU \equiv UD$, and then add the rules $C \leftrightarrow D$ to the Thue system. We can then apply these rules using R as a starting symbol to derive a further set of identity sequences $R_1, R_2, \ldots$. These, in turn, can be factored based on overlaps to provide a further set of rules $C_i \leftrightarrow D_i$ and so on. Since all R_i have the same length, as do all C_i and D_i , this process is guaranteed to terminate.

This is similar to, but not the Knuth-Bendix algorithm: there is no explicit concept of well-ordering, for example. However, it certainly contains many of the "basic features" of the algorithm as described by Buchberger (Buchberger 1987), and could be considered, under restrictive conditions, as an embryonic version of it.

References

Jean Berstel. Axel Thue's papers on repetitions in words: a translation. Publications du LaCIM, Université du Québec à Montréal, 1995.

Bruno Buchberger. History and basic features of the critical- pair/completion procedure. Journal of Symbolic Computation, 3(12):3– 38, 1987.

G.A. Hedlund. Remarks on the work of Axel Thue on sequences. Nordisk Matematisk Tidskrift, 15:148–150, 1967.

Emil L. Post. Recursive unsolvability of a problem of Thue. Journal of Symbolic Logic, 12(1):1–11, March 1947.

M. Steinby and W. Thomas. Trees and term rewriting in 1910: On a paper by Axel Thue. EATCS Bull., 72:256–269, 2000.

Axel Thue. Probleme uber Veränderungen von Zeichenreihen nach gegebenen Regeln. Christiana Videnskabs-Selskabs Skrifter, I. Math.- naturv. Klasse, 10, 1914. My (anonymised) translation of [Thu14] is available at: `http://tinyurl.com/thue1914-pdf`

Computers and obedience: defining machine autonomy in the 1940s

Mark Priestley (m.priestley@gmail.com)
UCL, United Kingdom

Historians have often noted that the new computing machines of the 1940s were frequently de- scribed in anthropomorphic terms, for example as robots or as "giant brains". By the end of the decade, this tendency had crystallized into an often impassioned debate around the specific question of whether machines could be said to possess intelligence, or the ability to think.

The purpose of this paper is to use the concept of obedience as a tool to analyze some aspects of this anthropomorphizing discourse. Drawing on material from logic, philosophy, and popular fiction as well as the history of computing, it will trace some of the ways in which the idea of obedience came to be associated with that of machinic agency, and how these ideas helped shape responses to the emergence of the automatically sequenced computer.

The first section of the paper examines the early robot stories of Isaac Asimov. Written in the first half of the 1940s, these stories represent an attempt to imagine robots as manufactured products embedded in complex industrial societies, rather than simply as fantastic figures, and to respond to contemporary anxieties about the role of machines in human society. Asimov encapsulated the rela- tionship between robots and human society in his famous three rules of robotics, the second of which explicitly states that robots must obey humans.

The early stories can be read as both an explanation and as an exploration of the three rules. View- ing them almost as a formal system defining the boundaries of acceptable robot behaviour, Asimov constructs a number of apparently paradoxical situations in which the outcome of the rules is not at all what might have been expected. The drama of the stories is provided by the human protagonists' attempts to explain the robots' behaviour in terms of the rules that determine it.

Interestingly, Asimov described it as a "mathematical certainty" that a robot would obey the three rules embedded in its positronic brain. The second section of the paper argues that Alan Turing's 1936 analysis of computability provides just such a mathematical analysis of the relevant aspects of human behaviour, and that the universal machine can usefully be described as a machine whose primary function is precisely to obey the instructions given to it. That this interpretation was central to Turing's own understanding of

the universal machine is demonstrated by an examination of a passage from his 1951 Programmer's Handbook for the Manch- ester computer, in which Turing gives a formal definition of what is meant by obeying a command. Interestingly, his account is internal rather than external: the effect of a successful act of obedience is a change in the agent rather than any necessary change in the outside world.

After this examination of contemporary notions of machinic obedience, the third section of the paper examines texts from the 1940s which describe the new computers and begins the task of identi- fying some of ways in which the notion of obedience played out in practice, including the following.

Class and social position A number of texts, particularly from Britain, attempted to locate comput- ers in relation to positions in the social hierarchy characterized by obedience. Naturally, these tended to be lower-status roles, and computers were often presented as ideally disciplined work- ers.

Initiative It soon became apparent, however, that compared with the computer, even the most dis- ciplined human worker was expected to demonstrate considerable initiative. In a formulation that was widely attributed to Ada Lovelace, it was said that computers could do only what they were told to do, and it became recognized that the instructions given to machines had to be to an unprecedented degree complete, and completely explicit.

Surprise However, as Asimov's robots had demonstrated, this turned out to be less straightforward than expected, and the capability of machines and programs to surprise their creators with the unexpected or unintended consequences of the instructions given to them was frequently noted.

Responsibility Despite being thought of as partially autonomous, however, computers were not held to be responsible for these unexpected outcomes. Errors of performance, both mechanical prob- lems and those arising from the instructions given to them, were not deemed to be their fault and did not affect their status as "obeying machines"; as Turing observed, obedience was an internal, not an external, property.

Tracing the contours of machinic agency is not simply an intellectual exercise. In a move reminis- cent of Hegel's dialectic of the master and the slave, human agents began to reconfigure themselves in response to the interaction with computers. This is clearly visible in connection with programming. Early treatments of errors in automatic computation did not envisage a category of "programming errors", and Maurice Wilkes famously remembered as an epiphany the moment when he realized that much of the rest of his life would be spent correcting the errors in his own programs.

The apparent difficulty of anticipating the consequences of a set of formally given instructions therefore led to a transfer of responsibility: rather than it being the worker's responsibility to carry out orders in an appropriate way, it now became the manager's responsibility to ensure that orders were given in such a way that slavish obedience would lead to a reasonable outcome. Managers themselves had to become more disciplined in order to make use of that most disciplined of employees, the computer.

The final section of the paper turns from the positive to the negative characterization of machine agency. Hoping to establish that the computer, while more than a machine, was sill less than a fully autonomous agent, considerable efforts were made to describe what computers lacked. Answers proposed to this question centered around the notions of intelligence and the question, "can machines think?"

Causality in concurrent systems

Federica Russo (f.russo@kent.ac.uk)
Center Leo Apostel, Vrije Universiteit Brussel
Brussels, Belgium
Silvia Crafa (crafa@math.unipd.it)
Università di Padova, Italy

In the terminology of computer science, concurrent systems identify systems, either software, hardware or even biological systems, where sets of activities run in parallel with possible occasional interactions. A simple example of concurrent system is the Internet, which can be thought of as a set of computers, each one computing its independent activity, that often communicate to exchange some information. A further example is the railway system of a country, where many trains travel sharing tracks in an ordered way so that two trains can move at the same time along different tracks, whereas a single track (e.g, a platform in a train station) can only be used by a single train at a time. Furthermore, the large number of activities carried on by a single human cell form a biological concurrent system that actually shares a number of similarities with the Internet.

Compared to sequential systems, where a single action is executed at a time according to a sequential algorithm, concurrent systems raise new complex issues dealing with the ordering of action executions, since independent actions can be executed in any order or simultaneously. As a consequence computer scientists resorted to the causal terminology to describe and analyse the relations between the system actions. However, a thorough discussion about

the meaning of causality in such a context has not been developed yet. We then ask precisely what causality means and how causal reasoning works in concurrent systems. We rely on a precise formalization of the systems under observation, distinguishing between formal languages to specify or to program a concurrent system, operational models describing their behavior and analysis techniques to prove system properties. In particular, we consider concurrent systems modeled in terms of Event Structures (Winskel 1982), where the causality relation is given explicitly as a primitive relation and the causal talk is recurrent. First we observe that these models are not intended to be used for causal discovery: instead of asking whether two events are related by a causal relation or not, which might be difficult or controversial, they take causal relations as primitive, i.e., as already decided, so to allow formal reasoning on them. However, the difficult problem is not completely eluded: given a system, an event structure must be correctly associated to the system so that the primitive causal dependences of the event structure actually agree with the system behavior. The definition of a correct and useful model for a given concurrent system is a lively research topic. Anyway, for software systems there is again no causal discovery to do; the debate generally amounts to the definition of a 'precedence relation' between system instructions.

Moreover, in event structures causality means quite generally dependence, whether temporal, spatial, or even causal dependence. It is then very different with respect to more traditional debates in the philosophy of causality, for instance production and mechanisms, independence, and causation by omission. In concurrent systems causal talk may then appear 'loose', or even unnecessary, as causality just involves here a 'dependence' component with no 'productive' component. But all this is just to say that, in spite of similarities of type of problems, concurrent systems seem to have different worries, like formal reasoning about (any kind of) dependencies and the study of independent/concurrent actions.

As far as analysis techniques are concerned, we focus on tools like trace analysis and fault diagnosis, where the causal model of the system turns out very useful to reason about the chain of events (the 'causes') that led to a specific system state, i.e. to an error. Interestingly, such a process involves counterfactual reasoning. Indeed, counterfactuals are often used to reason about causes and effects, specifically about what would have happened had the putative cause not occurred. The goal of a counterfactual is then to pick out the 'right' cause and we'll know that it did in case it holds true.

There exists a vast and controversial philosophical literature dealing with counterfactual validation that mainly focused on Lewis's account based on

possible-world semantics. We rather observe that operational models of concurrent systems can be effectively used together with the theory of Nicholas Rescher (2007), which is capable of making sense of counterfactual validation in a way that is logically precise and rigorous, and that is metaphysically parsimonious. More precisely, in order to validate a counterfactual, Rescher's approach for restoring consistency by prioritising information (a.k.a. MELF) is well suited to formal operational models, where we can always decide the priority of beliefs thanks to the clear distinction between 'facts' and 'laws'. Given the nature of event structures, for each pair of events, it is known what relation they stand in; consequently, all the 'laws' connecting all the pairs of events are known from the model. Counterfactuals can then be validated by combining salient laws into a well-constructed proof. Conversely, given that the model describes all the possible system executions, a counterfactual can be rejected by showing a case, namely a possible execution that violates it.

To conclude, the formalization of concurrent systems is an interesting area to investigate the meaning and use of causal concepts. The literature in computer science customarily uses causal terms, but a systematic investigation has not been carried out so far. The analysis above suggests that causal talk in concurrent systems diverges from the traditional meaning in the philosophy of causality: it may as well dispose of the term 'causality' and employ 'dependence' instead, without loss of content in its modeling practices. Yet, our goal is not to call for a terminological change in the field of concurrent systems. We think that at this stage a rounded discussion about similarities and dissimilarities with parallel debates happening in the philosophy of causality is already a contribution.

References

Rescher, N. (2007). Conditionals. The MIT Press.

Winskel, G. (1982). Event structure semantics for CCS and related languages. In: Automata, Languages, and Programming. LNCS, vol. 140, pp. 561–576. Springer.

Is Networking computing ?

Valérie Schafer (valerie.schafer@iscc.cnrs.fr)
CNRS, France
Francesca Musiani (francesca.musiani@gmail.com)
Mines Paris Tech, France
Benjamin Thierry (benjaminthierry@gmail.com)
Paris Sorbonne, France

"Network science" is currently developing as an academic discipline-of-disciplines. It parallels computer science, its developments, and forms interesting hybrids with research more closely related to computational problems. The semantic shift from computational to digital is increasingly commonplace, and has a French counterpart in the choice of the word numérique, which attempts to account for a phenomenon that goes beyond mere computing. In this context, critically raising the question "Is networking computing?" does not seem unfounded. It is even less so as, despite its somewhat provocative allure, it has been raised since the very beginning of the history of networks.

Thus, this paper will, in a first part, re-examine the questions that researchers asked during the networks' "first steps", in the 1960s and 1970s, on their perimeter, their definition, their role. Researchers in computer science and in telecommunications, in particular, find in networks a venue of dialogue, but interrogate themselves on the respective places of their fields of study in this convergence, which allows to think about the relation between networking and computing in a diachronic way. Investigating the status of network research in the decades preceding ours, and study the computing-networking nexus, also entails a closer look to the men, the institutions, the research and the implementations that interested them the most. It means to address the integration, or the marginality, of different research teams, their understanding of being part of the computing culture or of a new field-in-the-making. It implies a careful analysis of the position of other actors – such as, e.g. in research on packet switching, ATT or French CNET – as the positioning vis- à-vis these actors is used by researchers as a way to ultimately define themselves. And finally, it means to look more closely at hybrid objects and systems that are landmarks of this convergence, whether it's the Minitel, the Internet, or today's cloud.

Our initial question is, therefore, grounded in history – and interrogates historians themselves: are historians of networks computing historians? Arrived after computing historians, who started their work by focusing on periods of time in which data networks did not exist, network historians needed, all of

a sudden, to take into account the societal implications of their object, and could not avoid yielding to the "internalist" tradition (history of technology proposed by technical people) which had characterized, in its early days, a history of computing fascinated by the machine. Notwithstanding, this should not be taken as evidence of the fact that network historians have not learned from history of computing and from its epistemological and historiographical evolutions. These have witnessed, in turn, interest shifts from hardware to software (Ensmerger), to information society (Aspray, Castells), and to the "digital revolution" (Misa). Furthermore, the preservation of this link allows them to incessantly reconsider the materiality of networks – that, without neglecting the virtual, allows them to avoid a content-exclusive approach – and to remain anchored to the study of digital objects that relate to the histories of innovation, technology, media and enterprise. At the crossroads of several fields, the history of networks invites, as well, to reflect upon its specificities, those of its archives; how it adds to the history of computing while owing to it; with the aim, finally, to enrich the already-lively historiographical debates that the latter has been experiencing for several years.

Finally, the third part of this paper will address the cultural, social, political and juridical roots – beyond the technical and scientific ones – that have shaped networks, to try and single out how the imaginaires and practices of networking have evolved, in relation (and in opposition) to those of computing. To do so, we address themes such as openness, information and communication, languages, data and interfaces – themes that are common to both fields, and can weave interesting links, if we take into account, rather than computers or networks as objects, computing or networking as foundational dynamics of a field where research and practice are increasingly complementary – and intertwined.

References

Abbate J. (1999). Inventing the internet. Cambridge: MIT Press

Agre, P. (2003). Peer-to-Peer and the Promise of Internet Equality. Communications of the ACM, 46 (2) : 39-42.

Braman S. (2010). Technical Design of the Internet and the Law : The First Decade. `http://microsites.oii.ox.ac.uk/ipp2010/programme/111`

Ensmenger N. (2004). Power to the people: Toward a social history of computing. IEEE Annals of the History of Computing, 26 (1), 94-96.

Ensmenger N. (2012). The Digital Construction of Technology: Rethinking the History of Computers in Society. Technology and Culture.

Flichy P. (1999). Internet ou la communauté scientifique idéale. Réseaux, nÂř 97, vol. 17, 77- 120.

Flichy P. (2001). L'imaginaire d'internet. La Découverte, Paris.

Hauben R. et M. (1997). Netizens: On the History and Impact of Usenet and the Internet. Wiley-IEEE Computer Society Pr.

Mahoney M. (1998). The History of Computing in the History of Technology. Annals of the History of Computing, 10, 113-125.

Misa T. J. (2007). Understanding how Computing has changed the World. IEEE Annals of the History of Computing, 52-63.

Musiani F. & Schafer V. (2011), "Le modèle Internet en question (années 1970-2010)", Flux, 85-86 (3-4), pp. 62-71.

Moatti A. (2012). Le numérique, adjectif substantivé. Le débat, nÂř 170, 133-137.

Neff, G. & Stark, D. (2003). Permanently Beta: Responsive Organization in the Internet Era. In Howard, P. & Jones, S. G. (eds.) Society Online: The Internet in Context, Thousand Oaks, CA : Sage Publications, 173-188.

Paloque-Berges C. (2010). Entre trivialité et culture : une histoire de l'Internet vernaculaire. Emergence et médiations d'un folklore de réseau. Thesis in Information and Communication Sciences, Paris 8 University.

Quaterman J. (1989). The Matrix: Computer Networks and Conferencing Systems Worldwide. Digital Press.

Schafer V. & Thierry B. (2012). Le Minitel, l'enfance numérique de la France. Nuvis, Paris.

Serres A. (2003) Aux sources d'Internet : l'émergence d'Arpanet, Presses Universitaires du Septentrion, Villeneuve d'Ascq.

Rheingold H. (1996). Les communautés virtuelles. Addison-Wesley/France.

40 years of computer science PhD in Lille University

Yann Secq (Yann.Secq@univ-lille1.fr)
Université Lille I, France

C omputer science has emerged in France in the late fifties with the development of calculators usages within universities laboratories. Works on these calculators started within mathematical laboratories and more specifically in the applied mathematics domain of numerical analysis. Then, gradually new themes have emerged that were not tied to numerical considerations. These works have progressively led to the creation of computer science laboratory that have gained their autonomy from mathematical laboratories.

This study is focused on the evolution of PhD subjects from the creation of the *Laboratoire de Calcul* of the *Université de Lille* in 1958 until the end of 2012. We have categorized PhD subjects and traced their evolution during these 50 years from the beginnings of computer science PhD to the specialization stage that has exploded during the eighties. We also describes PhD supervisors training and their scientific genealogy to relate the development of given research categories.

We hope that this work can be seen as a contribution for the description of one community of the communities of computing as Mahoney (2005) call them. Finally, we try to confront what has happened within this laboratory to others studies done at a larger scale concerning the evolution of research themes in computer science history.

Method

This study is a part of an ongoing effort that has started in november 2012 to preserve the heritage of the computer science laboratory (*Laboratoire d'Informatique Fondamentale de Lille*, LIFL[1]). Two workshops involving retired colleagues have been organized to this aim. The first one in November 2012 was focused on the emergence of the *Laboratoire de Calcul* within the Lille University and the second one in May 2013 on the creation and developments of computer science diploma. A third one should happen in December 2013 during the commemoration of the 30 years of the LIFL. This work relies on a collective involvement of severals colleagues without whom this paper would not exist:

- Joseph Losfeld who has initiated this work by organizing the first meeting in november 2012,
- all retired colleagues that have positively answered our call to participation in this heritage preservation action,
- Pierre-Eric Mounier-Kuhn which was present at both workshop and has gently suggested to present this work in progress to the HaPoC community,
- Sylvie Moine and Isabelle Le-Bescond from our university library which have been helpful to build the exhaustive listing of all PhD thesis.

To sustain and foster the work done during the first two workshops, a wiki website[2] has been started to organize and to make available resources that have been gathered. All PhD notices are available on this wiki and can be downloaded in an easily usable format (CSV).

Results

[1] `http://www.lifl.fr`

[2] `http://wikis.univ-lille1.fr/scite/site/histoireinfo/home`

The results are based on an extraction of data available on the SUDOC[3] and ENS ULM RubENS[4] websites. Notices older than 1973 have been gathered on RubENS while others notices have been extracted from SUDOC.

The database contains 329 PhD notices ranging from 1967 until 2012. These notices are still being manually checked by colleagues to complete them because some fields are missing, particularly the director field.

Year	67	68	70	73	74	76	77	78	79	80	81	82	83	84
# of PhD	1	1	1	1	2	5	1	3	14	3	3	5	6	11

Year	85	86	87	88	89	90	91	92	93	94	95	96	97	98
# of PhD	4	5	1	7	9	12	11	8	10	14	10	9	13	16

Year	99	00	01	02	03	04	05	06	07	08	09	10	11	12
# of PhD	10	10	5	8	2	10	13	16	15	15	8	16	9	6

Kalmár's Argument Against the Plausibility of Church's Thesis

Mate Szabo (mszabo@andrew.cmu.edu)
Carnegie Mellon University, USA

In his famous paper, An Unsolvable Problem of Elementary Number Theory, Alonzo Church (1936) identified the intuitive notion of effective calculability with the mathematically precise notion of recursiveness. This proposal, known as Church's Thesis, has been widely accepted. Only a few people have argued against it. One of them is László Kalmár, who, in 1957 gave a talk in Amsterdam at the International Colloquium "Constructivity in Mathematics," entitled An Argument Against the Plausibility of Church's Thesis. The talk was published as (Kalmár 1959). The aim of this paper is to present and analyze Kalmár's argument in detail.

It is very useful to have an insight into Kalmár's general, sometimes peculiar, views on the foundations of mathematics; my discussion is based on his (1942) and (1967). According to him, mathematics not only stems from experience and empirical facts, but even its justification is in part empirical. The development of mathematics, mathematical methods and notions is endless.

[3] http://www.sudoc.abes.fr
[4] http://halley.ens.fr/search*frf S5

As a consequence, mathematics cannot have a fixed foundation once and for all. Finally, presenting mathematical results in given, fixed frameworks is useful for precision and clarity, but mathematics is always done on an intuitive level, not in one or many of these frameworks.

Kalmár considers Church's Thesis as a pre-mathematical statement: it cannot be a mathematical theorem or definition, as it identifies a mathematically precise notion with an intuitive one. Thus, his argument against the plausi- bility of the thesis is also pre-mathematical. Kalmár begins by discussing his understanding of effective calculability, which is less restrictive than Church's, and questions the "objective meaning" of the notion of uniformity. That allows him to draw some "very unplausible" consequences of the thesis. It "implies the existence of an absolutely undecidable proposition which can be decided." This proposition is absolute in the sense that it is not undecidable relative to a fixed framework as the Gödel sentence is, but it is only one proposition and not an infinite set of propositions as Churchs undecidable problems are. However, the proposition can be decided on an intuitive level.

Kalmár's different understanding of the notions of effective calculability and uniformity were not only motivated by his general views on the foundations of mathematics. His epistemological as well as his political views played a significant role in it. He expressed these views in his talks on the same topic in Hungarian in his (1952) and (1957). Within this broader context, Kalmár's rather short and peculiar paper appears a bit more appealing. However, in the end his argument does not affect Church's Thesis, given the usual understanding of effective calculability as mechanical procedures.

It is worth mentioning that Gödel's explanation of his incompleteness results and his conclusion that it "is not possible to mechanize mathematical reasoning" (193?) and some of Kalmár's arguments concerning Gödel's and Church's results resemble each other remarkably. Nevertheless, their stance on Church's Thesis is quite different. I will use the dissimilarities of their arguments to point out once again where Kalmár's takes a defective turn.

References

Church, Alonzo. 1936. "An Unsolvable Problem of Elementary Number Theory." American Mathematics 58, no. 2: 345–363.

Gödel, Kurt. 193?. "[Undecidable Diophantine Propositions]." In Kurt Gödel's Collected Works. Unpublished Essays and Lectures, Vol III, New York–Oxford: Oxford University Press, 164–174.

Kalmár, László. 1942/2011. "The Development of Mathematical Rigor from Intuition to Axiomatic Method." Translated by Zvolenszky Zsófia, In Máté András, Rédei Miklós and Friedrich Stadler (eds) The Vienna Circle in Hungary. Wien–New York: Springer–Verlag, 269–288.

Kalmár, László. 1952. "A matematika alapjaival kapcsolatos újabb eredmények (New Results Concerning the Foundations of Mathematics)." In A Magyar Tudományos Akadémia Matematikaiés Fizikai Osztályának Közleményei 2, no. 2: 89–112.

Kalmár, László. 1957. "Az ún. megoldhatatlan matematikai problém ákra vonatkozó kutatások alapj ául szolg áló Church-féle hipotézisrol (About Church's Thesis and Unsolvable Mathematical Problems)." In A Magyar Tudományos Akadémia Matematikaiés Fizikai Osztályának Közleményei 7, no. 1: 19–38.

Kalmár, László. 1959. "An Argument Against the Plausibility of Church's Thesis." In Arend Heyting (ed) Constructivity in Mathematics, Proceedings of the Colloquium Held at Amsterdam. Amsterdam: North– Holland Publishing Company, 72–80.

Kalmár, László. 1967. "Foundations of Mathematics – Whither Now?" In Lakatos Imre (ed) Problems in the Philosophy of Mathematics. Amsterdam: North–Holland Publishing Company, 187–207.

Back to the (Libraries of the) Future

Mirko Tavosanis (tavosanis@ital.unipi.it)
Universit à di Pisa - Dipartimento di Filologia, lett. e l., Italy

The role of J. C. R. Licklider as a pioneer in the creation of modern computing is well understood (Hiltzik 1999; Bardini 2000; Rheingold 2000). However, most of the pertinent research is related to the work Licklider carried out in the second half of the Sixties, mainly centered on the establishment of interfaces and networks and culminating in the seminal paper The Computer as a Communication Device (April 1968) written together with Robert Taylor, illustrated by cartoons of Roland B. Wilson and published only a few months before Engelbart's famous "Mother of All Demos" in California.

Comparatively less known is instead the wider scope of Licklider's aims. In the Fifties, keeping track of the way he was spending his working days as expert in psychoacoustics, Licklider realized that a good part of his time was occupied by simple mechanical tasks which did not require particular intelligence. In 1960 he then published the seminal paper Man-Computer Symbiosis, where he explicitly stated the importance of a direct interaction

between the human operator and the machine in order to improve knowledge work. In this article, among other things Licklider proposed the widespread use of three systems: a "desk-surface display and control" (a screen on which the operator to track characters and pen drawings that the computer interprets and "regularizes"), a "computer-posted wall display" for meetings and finally "automatic speech production and recognition". It is interesting to note that none of these systems resembles a modern computer.

Licklider believed then that in a reasonable time "most of the task (...) of any technical thinker would be performed more effectively by machines" (Rheingold 2000, p. 134). The idea that the computer could not replace the scientist, but his aides, at that time was already the focus of the work of other researchers, such as Engelbart. Licklider, however, soon found himself in a unique position that allowed him to take mighty steps toward the concretization of his idea: in October 1962 he became director of the Information Processing Techniques Office (IPTO) of the Pentagon, a structure that in practice was responsible for the allocation of funds from the U.S. Ministry of Defense to the computer industry.

In 1964, when he was also busy writing Libraries of the Future, Licklider left the post of Director of IPTO and recommended (successfully) the young Ivan Sutherland as its substitute. Sutherland, however, held the position for only a few months, and in 1965 he was replaced not by a computer scientist, but by another expert in psychoacoustics: Robert Taylor, who followed scrupulously the search paths set by Licklider, of which he was indirectly student (Hiltzik 1999, p. 15, in his new role, among other things, Taylor was one of the main supporters of the work of Engelbart in 1967: Bardini 2000, p. 23). Goals like the production of computers capable of time-sharing and of interacting through graphic displays, or as the creation of ARPAnet, the first embryo of the Internet, became then central in the funding policy of the IPTO, according to the guidelines set in Libraries of the Future.

In Licklider's vision, however, those bold moves were meant only as a section of a wider restructuring of knowledge work. From this more general point of view, Libraries of the Future constituted an eerily prescient description of some fields of study that were developed independently in the following decades. In the book, in particular, Licklider described in full detail the composition and the working ways of a "procognitive system" where real-time computer networks and sophisticated tools of input and output were linked to natural language processing tools and wider repositories of information classified in machine-readable format. Particularly interesting from this point of view is the detailed comparison between Licklider's idea and

Tim Berners-Lee's description of a "semantic web" (Berners-Lee, Hendler and Lassila 2001).

However, it seems that the success of some sections of the program carried also the seeds of oblivion for the overall idea. Licklider's framework was quickly forgotten and was not considered as a reference for the computing of the following decades. A quick survey of the state of the art shows instead, if not an ongoing relevance of the framework, at least its usefulness as benchmark for the progress of computing and for a better understanding of some recurrent misperceptions regarding the nature of the information used in knowledge work

References

Bardini, Thierry. 2000. Bootstrapping. Douglas Engelbart, coevolution, and the origins of personal computing. Stanford University Press, Stanford.

Berners-Lee, Tim, James Hendler and Ora Lassila. 2001. "The Semantic Web". Scientific American Magazine, 17 May 2001.

Hiltzik. 1999. Dealers of lightning. Xerox PARC and the dawn of the computer age. HarperCollins, New York.

Licklider, Joseph C. R. 1960. "Man-Computer Symbiosis". IRE Transactions on Human Factors in Electronics, HFE-1, pp. 4–11, March 1960 (Licklider 1990, pp. 1-20).

Licklider, Joseph C. R. 1990. The computer as a communication device. Palo Alto, Digital.

Licklider, Joseph C. R. and Robert Taylor. 1968. "The Computer as a Communication Device". Science and Technology, 1968. (Licklider 1990, pp. 21-41).

Towards a Semiotic Framework for Programming Languages

Andrea Valle (andrea.valle@unito.it)
Università degli Studi di Torino, Italy
Alessandro Mazzei (mazzei@di.unito.it)
Università degli Studi di Torino, Italy

A programming language is "an artificial formalism in which algorithms can be expressed. For all its artificiality, though, this formalism remains a language" (Gabbrielli and Martini 2010). Considering this property, Tanaka-Ishii motivates a semiotic analysis of the programming languages (Tanaka-Ishii 2010). It is the only large-scale sign communication system not intended

uniquely for humans: “The purpose is the communication of programs between computers, from man to computers, and also from man to man.” (Zemanek 1966). We can arrange programming languages considering their proximity to the machine (low-level) toward the human (high-level). In the artificial ecosystem created by the Von Neumann machine the principal referent in low-level languages is the processor, with its specific binary dialect. Just upon this level we can recognize assembler languages, where a basic linguistic representation is introduced by using symbolic name for operations (e.g. STORE, LOAD, ADD) and by using addresses in order to refer to memory locations. Moving up we find high-level languages, in which the abstraction allows for the introduction of a "structured" form of linguistic representation based on the notion of "control flow" by means of conditionals and loop (e.g. IF, WHILE). Here we find an osmosis between human and machine semiotics, where the strict formal correctness of the machine side is balanced by the variety typical of natural languages.

In programmers communities a number of good practices has emerged in relation to the crucial notion of ‘readability’ of the code. For instance, the so-called good practices prescribe the insertion of comments, i.e. natural sentences that are directed uniquely toward the ideal human-reader and that are eliminated in the compilation/interpretation process. Good practices prescribe the use of meaningful names for variables, where meaning depends on the ideal human reader.

All programming languages are Turing-complete. This means that all languages are able to express the same "things", but the variety of languages (thousands of languages in fifty years) demonstrates the need to express some of these ‘things’ better (more easily, more efficiently) than others. In other words, there is obviously a connection to the Sapir-Whorf hypothesis in the relationship between programming language and what it can express. The explicit linguistic nature of programming languages, although little investigated by current semiotic literature, make them an interesting test bed for a theory of enunciation, that takes into account the roles played by the concepts of persona, time and space. From ethnographic point of view, many famous programmers consider the linguistic abilities, in the sense of natural language, as a important prerequisite to become a skilled programmer (Seibel 2002). We briefly analyze the concepts of persona, time, and space in the imperative, functional and object-oriented paradigms.

Imperative paradigm is the older family of high-level languages: “... -imperative- here has to do with natural language: as in an imperative phrase, we say -take that apple- to express a command, so with an imperative command we can say ‘assign to x the value 1’ ” (Gabbrielli and Martini 2010).

Here the subject of the enunciation "I" is an abstract entity, the subject-programmer who get into a relationship with an asymmetric and complementary subject "YOU", an abstract agent of calculus which implements the given orders. The memory is the reference space of the language: the names of variables establish a system of real mnemonic loci, in the double meaning of memory addresses for the machine and "placeholders" for the human interpreter. Thus, in the imperative paradigm, there is a clear opposition between an active dimension of the subject (agentive) and a passive dimension of the data (spatial). With regard to temporal dimension, imperative languages do not provide references to past or future. Every statement prescribes an action to be realized at the time of its enunciation: the sequence of enunciates coincides with the advancement of time.

In the functional paradigm, "computation proceeds by rewriting functions and not by modifying the state" (Gabbrielli and Martini 2010). Usually we can separate a functional programs into two sections: in the first section we find a set of inter-related function definitions; in the second section there is a 'request' to the 'environment' (no specific 'YOU' is present here) to compute the output value of a function over a specific input value. The programmer constructs an 'imaginary geography' of functions that can partially or exhaustively be explored. In other words, the functions establish a space of possible relations, governed by a specific topology. This function space also absorbs the time: the first function-call is the trigger of a function-call tree into the space of the function. The basic assumption in the pure functional paradigm is that the order in which this graph is explored does not affect the final result of the computation.

The foundational assumption in the object-oriented paradigm is to represent the human conceptualization of the world. Following a classical Aristotelian perspective, the basic structure is a taxonomy of classes that organizes the world into objects, where an object is an entity with properties (its 'attributes') and abilities (its 'methods'). The program consists of two blocks: the first descriptive (metalinguistic) one which provides for the definition of classes; the second one in which the objects are invoked.

The typical object-oriented syntax has the shape name.method(arguments), i.e. "subject! Do this in this way!". So, the 'YOU' of the enunciation is not the calculation agent, but a plurality of possible receivers, i.e. the objects: this feature eliminates spatiality from the world, because data are encoded into object attributes. Considering time, the object-oriented paradigm provides a timeless description of classes (similar to functional paradigm) and moreover a 'call', i.e. a sequence of instantiations and methods calls (similar to imperative paradigm).

References

Gabbrielli M. and Martini S., 2010, Programming Languages: Principles and Paradigms, London, Springer.

Seibel P., 2009, Coders at Work, New York, Apress.

Tanaka-Ishii K., 2010, Semiotics of Programming, Cambridge-New-York, Cambridge- University.

Zemanek H., Semiotics and programming languages, Communications of the ACM (9-3), 1966.

Can Computing in Art Renew the Debate on Art?

Mario Verdicchio (mario.verdicchio@unibg.it)
Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Bergamo, Italy

To take Computing into account in the context of Art is anything but simple. The task is complex because there is a number of issues to consider and because such issues are entangled in an intricated web of mutual influence. This work is an attempt to shed some light on the matter, with the aim to bring home at least some clear perspective from which we can conduct the discourse on Computing in Art and, hopefully, some new insights into the nature of Art.

Fundamental questions like "What is Art?" and "What is Computing?" are still very controversial. The former points at a long standing debate (Davies 1991) whose scope was made even wider in the beginning of the 20th century by groundbreaking and controversial works like Duchamp's Readymades (Kuenzli and Naumann 1989). The latter might appear to point at a narrower context, because the range of computing devices seems to be more manageable than the vast variety of works of Art out there, but nevertheless Computing is not devoid of conceptual issues surrounding its definition as a discipline (Tedre 2011) or the criteria that qualify computing devices (Searle 1980, Block 1990).

The lack of a solid conceptual framework in either field is reflected in the several names with which scholars refer to their intersection: "digital art", "generative art", "interactive art", "computer art", "online art" are some examples. Each term focuses on a peculiar characteristic that is meant to differentiate Art made with computing devices from more traditional endeavors. In particular, theorists seem to present interactivity as one of the

defining characters for this type of Art (Carter & Geczy 2006, McIver Lopes 2010).

Interactivity is often said to redefine the role of the spectator, who acquires a kind of co- authorship by activating some prompts, although always within the constraints predefined by the artist. Still, interactivity alone is not sufficient to fully characterize Computing in Art: sculptures with a reflective surface like Kapoor's Cloud Gate in Chicago (Baume 2008) also give the viewer the possibility to alter what they see on the artwork. An obvious reply may be that a piece of stainless steel is passive, whereas a fully-fledged interactivity requires the artwork to include devices that actively respond to the stimuli from the viewer.

However, there is no physical or conceptual requirement that obliges the artist to create such active responders by means of computing devices: purely mechanical or electromechanical feedback systems, comprised of sensors and actuators, would perfectly work in terms of interactivity. There is a conceptual issue here: are mechanical devices to be considered performing computations because their behavior depends on external stimuli and, hence, they can be seen as information processing systems, although there is no symbolic encoding at work? Answering this question on the nature of Computing is no easy task, but it is not the main point when it comes to Art. Whether they are computing devices or not, why are purely electromechanical systems employed only rarely nowadays, whereas an overwhelming majority of interactive artworks rely on (less conceptually controversial) computers? There are operational considerations to be made.

Snibbe's interactive work "Boundary Functions" projects lines between people on a platform, defining the contours of their personal spaces; as the persons move the lines change, too (Snibbe 1998). Imagine realizing such work without resorting to a computer: it is not impossible but it would require a much bigger effort by the artist to design and build the whole installation, possibly involving weight sensors to detect the participants' positions on the platform instead of a computer-based image analysis of the input from a camera.

If it becomes a question of performance (whether it is the speed at which the artwork responds, or the completion time by the artist), are computers employed in Art today because they are computing devices or because they are technological devices that guarantee results within certain resource constraints?

Computing in Art today is not only about computation, but how such com-

putation is carried out: computing devices play an important role also thanks to the technology they are based upon. The final result that the viewers enjoy is not the only part of the whole artistic endeavour to exploit technology: the way artists work is indeed changed by the benefits provided.

The impact on the creation process has been traditionally underestimated in the philosophical debate on Art, especially if the discourse ends up in the functional vs procedural debate. In trying to define the role of Computing in Art it might be necessary to go beyond such dualism.

References

Baume, N., editor (2008). Anish Kapoor: Past, Present, Future, MIT Press.

Block, N. (1990). "The computer model of the mind", Thinking: An Invitation to Cognitive Science, 3, MIT Press, pp.247-289.

Carter, M. & A. Geczy (2006). Reframing Art, Berg.

Davies, S. (1991). Definitions of Art, Cornell.

Kuenzli, R. E. & F. M. Naumann, editors (1989). Marcel Duchamp: Artist of the Century, MIT Press.

McIver Lopes, D. (2010). A Philosophy of Computer Art, Routledge.

Searle, J. R. (1980). "Minds, brains, and programs", Behavioral & Brain Sciences 3(3), Cambridge, pp.417-457.

Snibbe, S. (1998). Boundary Functions. `http://www.snibbe.com/projects/interactive/boundaryfunctions`

Tedre, M. (2011). "Computing as a science: A survey of competing viewpoints", Minds & Ma- chines, 21, Springer, pp.361–387.

Computer art or art of computing? Early debates revisited.

Joanna Walewska (`joanna.walewska@gazeta.pl`)
Nicolaus Copernicus University, Krakau, Poland

Since its very beginning, computer art has operated on the margins of art establishment as it was created in research laboratories and universities by engineers. Artists using computers tended to keep distance from contemporary art, but at the same time they needed to define computer art and its place in relation to traditional art. The status of computer art as art and its position in relation to such trends like op art, kinetic art or conceptual art were still negotiated. It was debated whether such computer art features

as generativity, processuality, reactivity, interactivity and creativity in the approach to technology can be a basis to recognizing it as an autonomous trend in contemporary art. In my paper, I would like to invastigate a process of recognition of computer art not as an iconic art but as purely intellectual or conceptual form. I will analyze two cases: first, a debate on the pages of PAGE bulletin which took place in the 1970's and then, a 2010 text by Frieder Nake from 2010, in which he reconsidered the status of computer art from the prespective of almost 50 years of its history and presented a view that it was virtually "more" conceptual than conceptual art.

In October 1971, Frieder Nake, one of the pioneers of computer graphics, wrote an article concerning the future of computer art entitled 'There Should Be No Computer Art', which started a dispute on the pages of the PAGE bulletin (nr 18, October 1971). It was provoked by his statement that he would no longer make art using a computer, because "the repertoire of results of aesthetic behavior has not been changed by the use of computers." Nake protested against the use of the new medium to create conventional art works, suitable for hanging on the wall of a gallery or museum. According to Nake, computers should be use as a tool for the liquidation of art, and he described the artists, who used it as if nothing had changed, as âĂđtechnocratic dadaists". The leading computer artists, who perceived Nake as an author of an algorythmic, geometric works referring to Paul Klee, considered it as an insult. The responses to this, so to say, Nake's "manifesto" appeared in the next issue of the bulletin in which John Lansdown published his article Computer Graphics does not equal Computer Art. In the article Lansdown tried to prove that the Nake's statement was true in relation to computer graphics, but one could not evaluate all the branches of computer art in such a way. According to Lansdown, computer art should be understood more as a process than a material object as only the former helps discover its potential. He thought that at least three artists should be appreciated as their artworks could not have been made without a computer: John Lifton, George Mallen and Edward Ihnatowicz, whose sculpture was described by Lansdown as "computer- controlled, 'intelligent', responsive to its environment in a way which makes other Kinetic art works seem like a toys". This response was followed up by a body of articles written by leading artists, which appears in subsiquent numbers of PAGE, but it seems that this heated discussion was inconclusive. In 2010 Nake wrote an article called Paragraphs on Computer Art, referring to the text by Sol LeWitt, in which he explicitly emphasized purely intellectual, algorithmic and semiotic nature of computer art, which should not be called computer but more precisely art of computing. By presenting this discussion along with the recent text by Frieder Nake I would like to situate the early computer art within the het-

erogenous conceptual framework of modernist art and to show its relation to the avant-garde.

An historical perspective on informatics language and music composition

Jean-Marc Wolff (jeanmarcwolff@hotmail.com)
Ministère de l'Education Nationale, Paris, France

What was the impact of the development of computer science on music composition practices since the mid-20th century ? What are the links between computer programming and music composition?

From the 1957 asssembler language used by Music 1 to the current QuteCSound language and Openmusic software, we will discuss the genealogical succession of informatics languages and softwares used for music composition in 'art music' and the way both were implemented and used by computer engineers and composers, in a interactive and cross-fertilised movement.

In this study, we firstly consider the growth of an 'invisible college'. From a small community of music-loving computer engineers like Max Mathews and Lejaren Hiller (some of them like Jean-Claude Risset will eventually become composers), composers such as Iannis Xenakis, Pierre Boulez and Kaija Saarihao seeking solutions or inspiration in informatics languages, joined the community. Both composers and engineers contributing to a common work through their various competences and needs, a cross- fertilised movement arose from this intellectual symbiosis. This movement was reinforced by academic institutions such as the MIT, Stanford or Princeton and by research institutions such as the IRCAM. An 'invisible college' early springs up at the end of the 1960's, in a strong interaction with IT networks and leads to the rise of a specific 'epistemic community'.

More precisely, we will discuss how informatics shaped music composition and how related composition issues lead up to new intellectual and creative dynamics among this community. While studiing several piece of art (from Hiller, Risset, Xenakis, Saarihao, etc.) we will deeply focus on specific time dynamics of both informatics and music composition. Indeed lags and convergence of this two technical an esthetic historicities gave birth to creativity in this two fields.

This study is conducted by using several engineers' and composers' testimonies extracted from specific books, specialized music reviews and scientific conferences's proceedings such as the International Computer Music Confer-

ence, and secondary sources which conduct analysis of musical works, with a specific emphasis of those from composers who worked at the IRCAM.

References

Dean R.T. dir.(2009), The Oxford Handbook of Computer Music, Oxford University Press, New York

Collins N., d'Escrivan J. ed. (2007), The Cambridge Companion to Electronic Music, Cambridge University Press, Cambridge

Computer Music Journal, since 1977

Leonardo since 1968 ; since 1991: Leonardo Music Journal

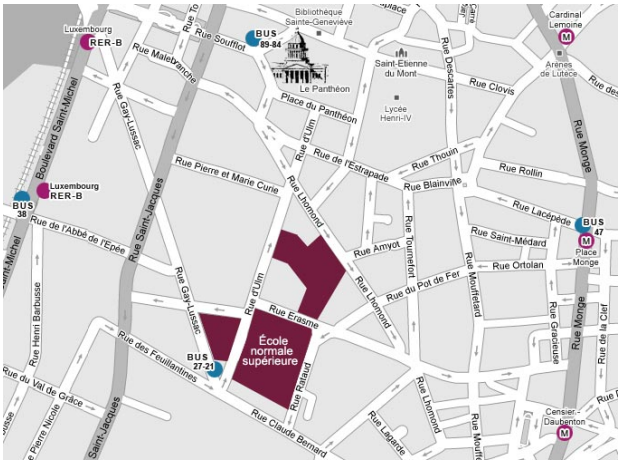
International Music Computer Conference (ICMC) since 1975

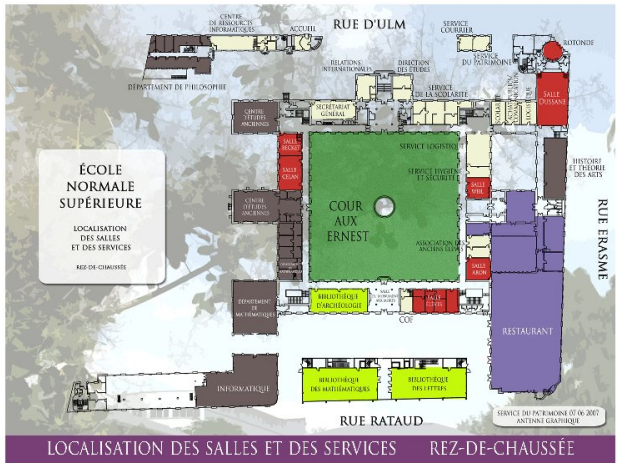
Journées Informatique et Musique (JIM), since 1994

Conference Venue

The conference will take place at the Ecole Normale Supérieure (ENS), rue d'Ulm 45, salle Dussane.

The afternoon session on Wednesday, 30th of October and the conference dinner that same afternoon will take place at the Amphithéâtre resp. la Rotonde of the *Ecole Nationale Supérieure des Arts Décoratifs*, rue d'Ulm 31.





Index