Functions, Abstractions, and Idealizations in the Explanation of Computing Systems' Behaviours

Nicola Angius (joint work with Guglielmo Tamburrini)
University of Sassari, Italy

Computing systems (CS) form a vast class of physical systems, embracing general purpose personal computers, myriads of special purpose computing devices embedded into many technological systems (cars, smartphones, robots), and worldwide networks of computers that cooperate to carry out concurrent or parallel computations. One of the main activities computer scientists are engaged into is evaluating the realized artefacts with respect to users requirements. Accordingly, explaining computing systems' behaviours (CSB), both correct and incorrect, is a pervasive and varied activity in computer science. However, the methodological problem of analyzing what it is to explain CSB has not received much attention in the philosophy of science and technology. This paper explores the problem of CSB explanations by focusing on machines belonging to restricted subclasses of the broad class of computing systems, i.e. digital computers running offline a variety of programs written in some high-level programming language. These case studies enable one to highlight significant features of explanations in computer science, concerning both correct and incorrect behaviours of those computing systems.

It is pointed out here that many explanations of CSB, which are acknowledged as adequately addressing the explanatory requests from which they arose, bottom out without making any reference to physical components and processes of computing systems. Accordingly, these explanations rely on purely *functional decomposition strategies* [2, 3] while abstract away from *all* physical components of CS and physical descriptions of the processes they engage into. This analysis of CSB explanations is brought to bear on mechanistic models of explanation [6]. The latter often come with the regulative ideal of the full physical instantiation of functional roles as a means to achieve greater explanatory force [7, 4]. However, following this regulative ideal does not invariably lead to better explanations in computer science, insofar as the functional decomposition strategy can be decoupled from the functional role filler instantiation strategy without losing in explanatory force.

Such thesis is here maintained by introducing two why-questions concerning CSB:

a) Why has digital computer *C* displayed an incorrect/correct behaviour in these particular executions (or runs) of program *P* on inputs I_1, ..., I_n?
b) Why were physically heterogeneous digital computers D_1 and D_2 capable of running the same program *P*?

User specifications usually tell programmers something about *what* is to be accomplished without saying much about the processes by means of which this is to be done. If some user specification admits a computational solution at all, then it can be fulfilled by means of an infinite number of programs written in many different programming languages. By selecting some particular high-level programming language *L* for writing a program, programmers introduce constraints on the *how*, on the means to be used to fulfil user intentions. CSB explanations fulfilling explanatory requests of type a) and b) are in this paper addressed by reference to hierarchically organized layers of nested what-how descriptions one usually finds in computer architecture manuals. At the bottom there is the device level, that is, the physical level whose main objects of interest (the "what" of the physical level) are transistors or other physical devices which can be used as switches. Immediately above the device level there is the logical gate level, which is concerned with logic gates that are built out of transistors or other physical devices functioning as switches. Further up in the hierarchy the microarchitecture level includes a collection of registers that form a local memory and a special circuit called the Arithmetic Logic Unit (ALU). Still ascending in the hierarchy one finds the Instruction Set Architecture (or ISA) level. Programs written in various high-level languages are translated in the form of ISA level instructions, going through the previous translation of assembly language instructions.

The examination of question (a) allows this paper to exhibit how functional analyses decoupled from mechanist instantiations are satisfactorily capable of supplying explanations answering this questions type. In the case of incorrect CSB, this is shown by advancing explanations of a machine failing in yielding the correct factorial *n!* for the input number *n*. Explanations are there provided by isolating functional what-layers, in the how-what hierarchy, that do not fulfil upper how- prescriptions. And in the case of correct CSB, explanations are here supplied by abstract and idealized computational models used in the formal verification method known as Model Checking [1] to check whether P satisfies given behavioural properties for each system's potential run. Question (b) concerns the explanation of chains of events that are observed across different runs of P, no matter whether correct or incorrect, on architecturally dissimilar CS. This paper shows that even in the explanation of P-commonalities of architecturally heterogeneous physical systems there is no need to take into account all the physical details of the system in order to achieve better explanations. On the contrary, distortive idealizations [5] from physical processes of each system are essential to adequately address explanatory requests of this sort. This latter point will be maintained on the basis of the examination of general specifications of discrete state machines that can be attributed to Turing's [8] analyses of computational processes.

References

[1] C. Baier, J.P. Katoen, *Principles of model checking*, MIT Press, 2008.

[2] W. Bechtel, R.C. Richardson, *Discovering complexity*, Princeton University Press 1993.

[3] R. Cummins, "Functional analysis", *Journal of Philosophy* 72(20), 1975, pp. 741-765.

[4] D.M. Kaplan, "Explanation and description in computational neuroscience", *Synthese* 183(3), 2011, pp. 339-373.

[5] E. McMullin, "Galilean idealization", *Studies in History and Philosophy of Science Part A* 16(3), 1985, pp. 247-273.

[6] G. Piccinini, "Computing mechanisms", *Philosophy of Science* 74(4), 2007, pp. 501-526.

[7] G. Piccinini, C. Craver, "Integrating psychology and neuroscience: Functional analyses as mechanism sketches", *Synthese* 183(3), 2011, pp. 283-311.

[8] A.M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", *London Mathematical Society* 2(43), 1936, pp. 230-265.