Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Computation as social agency: What, how and who

Johan van Benthem [a,b,c,*]

[a] University of Amsterdam, The Netherlands
[b] Stanford University, United States
[c] Tsinghua University, China

ARTICLE INFO

Article history:
Received 1 March 2016
Available online xxxx

Keywords:
Computation
Logic
Epistemization
Gamification

ABSTRACT

Computation today is interactive agency in social networks. In this discussion paper, we look at this trend through the lens of logic, identifying two main lines. One is 'epistemization', making computational tasks refer explicitly to knowledge or beliefs of the agents performing them. The other line is using games as a model for computation, leading to 'gamification' of classical tasks, and computing by agents that may have preferences. This provides ingredients for a fundamental theory of computation that shifts from what is computed to how it is computed and by whom, moving from output to social behavior. The true impact of this shift is not in learning how to replace humans, but in creating new societies where humans and machines interact. While we do not offer a Turing-style account of this richer world, we discuss what becomes of three classical themes: the Universal Machine, Church's Thesis and the Turing Test.[1]

1. Introduction: computing, from 'what' to 'how' and 'who'

The Turing Centenary Year 2012 saw lively debates about the nature of computing. Are we on the threshold of new styles of computation that transcend the limitations of established paradigms? Let us briefly recall three classical themes from the golden age of the 1930s and 1940s when a founding generation made computation a subject for mathematical inquiry, and in its wake, for practical development. First, by analyzing the bare basics of human computing, Turing defined a *Universal Machine* that can compute the result of any algorithm on any input, when both are presented in suitably encoded form. Subsequently, this notion supported the development of Recursion Theory, bringing to light both the basic structures and power of effective computation, but also its limitations as shown in the undecidability of the Halting Problem. Second, on the basis of Turing's model and equivalent ones proposed independently at the time by Church, Gödel, Post and others, *Church's Thesis* then claimed that all effectively computable functions over the natural numbers, a canonical domain that can mimic non-numerical computation by various encodings going back to Gödel and others, must coincide with the recursive functions, that can be computed on Turing machines. Finally, as the power of this paradigm became clear, it was suggested in the famous *Turing Test* that computation might well emulate human cognition, to the extent that in conversation, humans would not be able to tell whether they are interacting with another human or a machine.

* Correspondence to: Institute for Logic, Language and Computation, University of Amsterdam, P.O. Box 94242, 1090 GE AMSTERDAM, The Netherlands.
E-mail address: Johan.vanBenthem@uva.nl.

[1] This is a position piece about some current directions and issues at the interface of logic, computer science and other fields. We do not cover all aspects of agency, an area too vast for easy summary. We provide quite a few references, but have not attempted a complete bibliography.

http://dx.doi.org/10.1016/j.ic.2017.09.009
0890-5401/

Now, 80 years later, computer science and information technology have transformed the world of both machines and humans in unpredictable ways. Given the experience of this period, can we go beyond the bounds of the classical age? One way in which this issue has been framed is as one of extension: can we compute or solve a larger class of functions/problems after all? There are lively debates on this theme, with proposals like using infinite machines or letting the physical universe do computing in new ways ([19], [18]). Personally, I see no evidence in these debates that we can compute more than before, forcing us to extend the calibration class of recursive functions. That austere but powerful abstraction still holds good. But there is another line of extension, running in another dimension. One should make a distinction between the following two issues:

What can we compute? and *How can we compute it?*

This distinction is obvious in computer science, and accordingly, beyond Church's Thesis lies a less definitely settled question: what are natural styles of computing? Or if you insist on 'what' questions after all: do not ask what is computable, but what is *computing*, seen as a kind of process producing various forms of behavior, from publicly observable signals to private internal choices. Right from its start, the history of computer science has shown a wealth of ideas on these themes, and sophisticated perspectives continue to appear.

This paper is in the 'how' line: and it even adds an aspect of 'who' is doing the computing. This leads to a view of computation as a form of social agency. This view mixes ideas from computation with the study of human agency, looking at influences going both ways. I am not claiming that this social perspective is new, and I will in fact refer to various authors who have contributed to this line. But my discussion does aim at a systematic perspective on these trends from the viewpoint of logic, and I hope to point out interesting new questions and directions in this way that may not be common knowledge. Still, what follows are personal views based on my own work at these interfaces ([79], [80], [81]), and I do not claim to represent public opinion in either computer science or logic.

2. Computer science, some fundamental lines

For a start, here is some quick history. It will be familiar to computer scientists, but it is not on the radar of many logicians and philosophers, who tend to think of computer science as the systems help desk of the university, not as a supplier of fundamental ideas of broad intellectual relevance.

Logic and fine-structure of computing Turing machines have opaque programs stating in complete detail what to do, making heavy use of that old enemy of perspicuity called 'go to' instructions, [20]. Computer science took off when machine-independent programming languages were invented, with higher-level perspectives. Programs in such languages are like algorithms describing the essence of computational tasks at a higher abstraction level. Different programming languages have their own views, often drawing on traditions from logic. Thus, imperative programs in Algol or C^+ are like dynamified logical formulas, telling a machine what sort of results to achieve given certain preconditions. Such systems lend themselves to model-theoretic semantics in a usual style (first-order, modal, or otherwise), witness Hoare Calculus or Dynamic Logic, [39]. On the other hand, functional programming languages such as LISP or Haskell, akin to systems of lambda calculus and type theory, link to proof-theoretic and category-theoretic traditions in logic. And beyond these there are yet further paradigms: such as object-oriented programs, logic programs, and so on.

The experience gained in this way has a significance beyond computer science. The notion of algorithm occurs in many disciplines, and in its wake came further ideas whose impact is still spreading, such as the crucial importance of computational complexity, or the natural tandem between algorithms and data structure. These ideas can be seen at work in logic, cognitive science, and even in philosophy.

Distributed computing and process theory A new development around 1980 was reflection on the practice of distributed computing emerging at the time. One major line, moving up the abstraction level beyond programming languages, was the invention of Process Algebra by Hoare, Milner, Bergstra, Klop and others, [9], an abstract view of processes as graphs modulo some suitable notion of structural behavioral invariance, often some variant of modal bisimulation, [10]. While it seems fair to say that no broad consensus has emerged on one model of distributed computation, comparable in breadth of allegiance to Turing machines, a deep process theory did emerge with many features that have no counterpart in the classical theory of sequential computing, [96].

Again, general ideas are at work here. Parallel action and distributed processing occur everywhere, being the essence of biological and social life. But also, the methods by which these phenomena are studied are of general interest. Finding natural invariances as a basis for process theories is a key idea in mathematics, and it is still percolating further. And also, compositionality in design is a shared maxim in many disciplines today.

Process theories in computer science are legion. A widely used and vigorous old-timer is automata theory, [33]. A conspicuous newcomer is co-algebra, a theory of computing on infinite streams, tied to fixed-point logics and category theory,

Please cite this article in press as: J. van Benthem, Computation as social agency: What, how and who, Inf. Comput. (2017), http://dx.doi.org/10.1016/j.ic.2017.09.009