The Turing Machine as a Boundary Object: Sorting Out American Science and European Engineering[1]

Edgar Daylight & Erhard Schüttpelz

School of Media and Information

Siegen University

Herrengarten 3

57072 Siegen

Deutschland

Keywords

halting problem, history of computer science, Wittgensteinian critique on "science"

Abstract

We propose to consider the "Turing machine" as a *boundary object* in sociotechnical terms for two reasons.[i] First, computer scientists have defined the "Turing machine" in different ways. Some players characterized the machine with a one-way infinite tape, others preferred infinity in two directions. In both cases, the infinity involved has been taken to be an actual infinity by some and a potential infinity by others. Likewise, the workings of the machine have been defined with quadruple notation in certain books and with quintuple notation in others. Second, despite such mathematical variability, there is immutable content: each textbook definition adheres to the neo-Russellian tenet, that, *everything a computer (or a physical object in general) can do, a Turing machine can do as well*.[ii] The tenet conveys a computational version of logicism,[iii] which came to prominence in the 1950s with the writings of a first generation of computer scientists.[iv] The following 1958 words of John Carr illustrate this uptake of Turing machinery in connection with Artificial Intelligence:

> Based on Turing's proof about universal machines:
>
> 1. Living organisms can be abstractly defined as symbol manipulator.
> 2. Actions of living beings can be described by a program.
> 3. Digital computers have all the features of Universal Turing Machines.
> 4. Digital computers can duplicate human beings.[v]

Our two observations, concerning the *plasticity* and neo-Russellian *robustness* of the Turing machine, adhere to the 1989 definition of a boundary object.[vi] Moreover, a

[1] Both authors are financed by SFB 1187 "Medien der Kooperation" (Siegen University) and the first author was also financed by ANR-17-CE38-003-01 "PROGRAMme" (Lille University). This abstract has been accepted for presentation at the 11th British Wittgenstein Society Conference: *Wittgenstein and AI (2022)*, https://www.nchlondon.ac.uk/wittgenstein-ai-programme/

boundary object has *interpretive flexibility*.[vii] And so does the Turing machine in the context of computer programming, as we shall illustrate in the next paragraph — and more elaborately in our presentation — with an American and a Dutch reception of the unsolvability of the halting problem (of Turing machines).

Around 1967, Marvin Minsky and Edsger Dijkstra welcomed the neo-Russellian tenet that *everything a computer can do, a Turing machine can do as well*. Both men shared a common identity on opposite sides of the Atlantic Ocean, which benefited academic discipline building. Yet they positioned engineering in relation to Turing-machine theory differently. According to Minsky, mathematical theorems about Turing machines are rules that dictate the dos and don'ts of the engineer, including impossibility results (which hinge on an infinite abstraction of real machinery).[viii] While for Dijkstra, software engineering itself was a pure science. According to him, theoretical insights about Turing machines could, at best, advise the engineer.[ix]

Examining the interpretive flexibility of the Turing machine as a boundary object sheds light on the contradistinction between American "science" and European "engineering," and between natural laws and governing laws. The latter dichotomy was also contentious 30 years earlier, e.g., in an exchange between Alan Turing and Ludwig Wittgenstein in 1939, and remains so today in the philosophy of computer science.[x] Our historical findings will help software scholars compare intellectual cultures inside computer science, spanning multiple decades up till 2022.

---

[i] Susan Leigh Star, "This is Not a Boundary Object: Reflection on the Origin of a Concept," *Science, Technology, & Human Values* 35, no. 5 (2010): 601.

[ii] Scott Aaronson, *Quantum Computing Since Democritus* (Cambridge: Cambridge University Press, 2013), xxviii; Michael Sipser, *Introduction to the Theory of Computation* (Boston: Thomson Course Technology, 2006), 137; David Harel, *Algorithms: The Spirit of Computing* (Reading, MA: Addison Wesley, 1992), 233.

[iii] Andrew David Irvine, "Bertrand Russell," *The Stanford Encyclopedia of Philosophy (Summer 2017 Edition)*, Edward N. Zalta (ed.), https://plato.stanford.edu/archives/sum2017/entries/russell/

[iv] Edgar Daylight, "Towards a Historical Notion of 'Turing — the Father of Computer Science'," *History and Philosophy of Logic* 36, no. 3 (2015): 205-228.

[v] John Carr, "Languages, logic, learning, and computers," *Computers and Automation* 7 (1958): 21-22, 25-26.

[vi] Susan Leigh Star, James Griesemer, "Institutional ecology, 'Translations' and Boundary objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology," *Social Studies of Science* 19, no. 3 (1989): 393.

[vii] Susan Leigh Star, "This is Not a Boundary Object: Reflection on the Origin of a Concept," *Science, Technology, & Human Values* 35, no. 5 (2010): 601.

[viii] Marvin Minsky, *Computation: Finite and Infinite Machines* (Englewood Cliffs: Prentice Hall, 1967), 153.

[ix] Edsger Dijkstra, "EWD 316: A Short Introduction to the Art of Programming," Technical report, November 1974, www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD316.html; Edsger Dijkstra, "EWD 372: A simple axiomatic basis for programming language constructs," Technical report, 8 May 1973, www.cs.utexas.edu/users/EWD/ewd03xx/EWD372.PDF; Edsger Dijkstra, "EWD 869: Ter afsluiting van de "Inleiding tot de Kunst van het Programmeren"," Technical report, 7 Dec. 1983, www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD869.html

[x] Edgar Daylight, "Addressing the Question "What is a Program Text?" via Turing Scholarship," *IEEE Annals of the History of Computing* 43, no. 4 (October-December 2021): 87-91.