Reflections on Turing's approach to modelling states of mind

Meurig Beynon
Computer Science. University of Warwick, Coventry CV4 7AL

In discussing Turing's seminal 1936 paper, the Stanford Encyclopedia of Philosophy [22] highlights the way in which his conception of the 'Turing Machine (TM)' was guided by the idea of *modelling states of mind*: " ... in a bolder argument, the one he placed first, he considered an 'intuitive' argument appealing to the states of mind of a human computer. [17, p. 249]. The entry of 'mind' into his argument was highly significant, but at this stage it was only a mind following a rule."

There are two respects in which the subsequent treatment of TMs in mainstream computer science has sidelined 'intuitive' elements:

- the TM is viewed as the fundamental mathematical abstraction in a theory of computer science that is based on "computational thinking".

- computer science has promoted the computational theory of mind, which proposes that everything the human mind does is attributable to *following rules*.

In combination, these two viewpoints promote a narrow conception of computer science; they respectively root computing in an abstract machine model with a strict formal semantics, and set out to show that such a model is enough to give a good account of its core applications.

Throughout the short history of computer science, it has suited the political purposes of an emerging discipline to emphasise its connections with Turing's work - one of the supreme intellectual achievements of the twentieth century. But perhaps it is not so clear that computer science as currently conceived truly fulfils Turing's aspirations for a science of computing.

As Hodges points out in [11], Turing's vision was broader than that of a mathematical logician: "The essence of Turing's achievement was the discovery of a concept *with an application to* logic, rooted in ideas which lay outside mathematics" and, unlike 'Church's Thesis', "Turing's definition [of computability] was modeled on what human beings could actually do" [11]. Also, as Hodges goes on to observe: "[After 1948] Turing did ... surprisingly little ... to build up the modern science of computation" [11].

As for Turing's stance on the contribution that computational abstractions make to our understanding of mind, there are many clues in his discussion of "The Imitation Game" in [19]. It is clear that Turing sees the isssue of whether an interrogator might be deceived into attributing a machine's responses to a human as a *limited* question, and not one that decisively illuminates the nature of mind. For instance, he acknowledges that other concerns about the mind might be beyond the scope of his enquiry: "I don't wish to give the impression that I think there is no mystery about consciousness ... [b]ut I do not think these mysteries necessarily need be solved before we can answer the question with which we are concerned in this paper." Even in relation to his contention that a computer could succeed in the Imitation Game [19], Turing is careful to point out that - despite the rebuttals he gives in

response to objections to this possibility: "The reader will have anticipated that I have no very convincing arguments of a positive nature to support my views."

Hodges [10] refers elsewhere to the "trenchant materialist and atheist Turing who emerged after 1936", and it is perhaps these characteristics that have been most emphasised in subsequent building upon his legacy. In that context, it may seem surprising that, in rebutting counter-arguments to the idea that a computer might successfully play "The Imitation Game", Turing remarks: "[The Argument from Extrasensory Perception] is to my mind quite a strong one" [Turing, 1950]. But this is more explicable if, as Hodges advocates in [10], "we recognize [in Turing] the common thread - a great seriousness about the sheer mystery of mental phenomena, and an equally serious conviction that they must be reconciled with a scientific world view."

As the Stanford Encyclopedia of Philosophy quotation above emphasises, Turing's treatment of states of mind was conceived with a view to modelling "a mind following rules", as was appropriate for addressing the *Entscheidungsproblem*: a Turing machine should model procedures "definite in the sense that at every stage a completely explicit 'note of instructions' could be written down explaining what was to be done in such a way that another person could take up the work" [10]. Beyond question, Turing's insight informed the practical contributions he made to the development of computer programming. For instance, it gives a deep meaning - deeper than a programmer needed to appreciate - to the statement with which he prefaced his *General Remarks on Electronic Computers* in his Manual for the Ferranti Mk. I computer [20]: "Electronic computers are intended to carry out any definite rule-of-thumb process which could have been done by a human operator working in a disciplined but unintelligent manner." It also led him to identify the importance in programming of more disciplined use of mathematical notation [18].

But - to take nothing away from the extraordinary fertile nature of Turing's insight, and what has been, and can yet be, achieved within the framework of computational models in many disciplines - it seems likely that Turing himself did not consider the broader issue of the nature of mind to be closed. To quote Hodges [10] once again: "... we cannot feel that Turing had arrived at a complete theory of what he meant by modelling the mental functions of the brain by a logical machine structure". Significantly, in the spirit of an applied mathematician, Turing gave high priority to squaring mathematical theory with empirical evidence and practical applicability. In this connection, Hodges [11] contrasts Turing's approach to enhancing discrete computational models as models of physical systems through introducing randomness with "that of some modern theorists, who seek to outdo discrete computation by exploiting the very elements that Turing made little of" and whose approaches lead to difficulties that render them "meaningless without stability and robustness in the face of infinitesimal phenomena". As further testimony to Turing's practical orientation, we need only consider the degree of direct involvement he wanted in building early computers, his contributions to code-breaking and speech recognition, and his concern to take account of the inevitability of operational error in theorising about computers [11].

It may be that, by imputing greater authority to mechanical models of mind than Turing himself would have decisively endorsed, today's students of computer science are in danger of identifying mathematics with a blind process of inference such as can be carried out by a machine. Such a concern is raised by Byers [5], for instance, in motivating his account of mathematical practice. Turing's biographers describe him as a mathematician "[whose] native style was rough-and-ready and prone to minor errors" [7] for whom the notion of 'intuition'

had a particular fascination (cf. his mathematical work on the Riemann hypothesis [10]). If this seems to be at odds with the demystification of effective procedures that he achieved in his own work, an instructive parallel may be drawn with his older contemporary Emil Post, who concluded his paper 'Absolutely unsolvable problems and relatively undecidable propositions' (written in 1941, but only published posthumously [15]) by expressing his amazement at the reaction to Godel's undecidability results in the following terms:
"... mathematical thinking is, and must be, essentially creative. It is to the writer's continuing amazement that ten years after Gödel's remarkable achievement current views on the nature of mathematics are thereby affected only to the point of seeing the need of many formal systems, instead of a universal one. Rather has it seemed to us to be inevitable that these developments will result in a reversal of the entire axiomatic trend of the late nineteenth and early twentieth centuries, with a return to meaning and truth."
To the end of his life, - in the spirit of Post's injunction, - Turing seems to have been motivated to seek significance beyond an abstract logical interpretation for his TM concept. In realising his vision for computation he felt the essential need to establish the connection with physical reality. In [10], Hodges cites Penrose's summary of Turing's position in 1950: 'It seems likely that he viewed physical action in general - which would include action of a human brain - to be always reducible to some kind of Turing-machine action'. As Hodges later goes on to relate [10], this was to be an unresolved problem for Turing, who recognised the challenge presented by the indeterminacy principle in quantum mechanics.

It is hard to imagine how the academic discipline of computer science would have emerged without Turing's contribution. In his work, Turing showed extraordinary prescience in relation to many aspects of the "computer programming" related activity that has been the central focus of academic computer science throughout its history. But, at the time of his death, the transformative impact of computers and programming could hardly have been predicted. And in the same way that mathematics demands a broader account than formal systems can supply, so too does contemporary computing. In concluding this brief review, it is appropriate to look at ways in which modern computing and the science of computing to which we must now aspire is influenced by other perspectives on modelling human states of mind. This conclusion reflects the author's own research interest, under the auspices of the Empirical Modelling (EM) project [23], in seeking a broader alternative conceptual framework for computing.

In [19], Turing asserts that the problem of developing a digital computer that can succeed at The Imitation Game "is mainly one of programming". In the context of modern software development, it has become apparent that "the problem of programming" cannot be understood in a narrow sense. One of the most critical aspects of software development is that of binding meanings to artefacts that ostensibly are - or are to be - specified purely in computational terms. The idea that such semantic considerations can be comprehensively addressed by formal computational semantics has been criticised by reviewers representing many different perspectives. They include the expert software consultant Michael Jackson [12], the distinguished computer scientist Peter Naur [14] and the philosopher Cantwell-Smith [6]. The common theme in these, and other critiques, is that formal semantics can only go so far in mediating meanings in the software development process, especially where the activity involves "radical design" (cf. [12,21]).

Whereas it suited Turing's purpose in addressing his mathematical objective in [17] to consider human states of mind associated with carrying out a calculation, quite different kinds of states of mind feature in modern software development involving radical design.

Such a development process has to take account of the perspectives of many human participants whose understanding is mediated in quite different ways from those of the traditional "human computer": they cannot be expected to appreciate the full purposes or context for actions, to be able to interpret formal notations reliably, or to be able to communicate their wishes abstractly without reference to actual experience that can only be had, and skills that can only be developed, (e.g.) by interacting with a prototype system. And even though the functional goal and the process itself may be clearly specified, the practical situated knowledge needed to enact the process may itself be difficult to access - as when we try to make a pot of tea in a neighbour's house, and have to contend with finding the ingredients ('where are the tea bags?'), identifying the utensils we need ("is that a teapot?"), and determining how to configure these ("where do I plug this in?"). In developing software for reactive systems, this exploratory activity may take yet more extreme forms: in configuring devices and tuning their responses, it as if we are investigating the feasibility of constructing the very hardware on which our programs are to be executed [2].

The duality that separates "the given already engineered computing device" from "the to-be-specified abstract sequence of instructions to be performed on the device" is characteristic of the computational framework within which Turing was reasoning. Turing exploits this characteristic when he identifies The Imitation Game as "drawing a fairly sharp line between the physical and the intellectual capabilities of a man", and stipulates that "the interrogator cannot demand practical demonstration" [19].

An instructive comparison can be made between Turing's approach to modelling the mind of a human computer and that conceived by David Gooding [8] in his account of Faraday's seminal experimental work on electromagnetic phenomena. In interpreting the way in which this activity was conducted, Gooding [9] introduced the notion of 'construals' as "proto-interpretative representations which combine images and words as provisional or tentative interpretations of novel experience ... [that is] being created ... through the interaction of visual, tactile, sensorimotor and auditory modes of perception together with existing interpretative concepts including mental images". Such construals can be regarded as a means to knowledge representation in spirit similar to that advocated by Rodney Brooks (cf. [3] and [4]). In [9], Gooding invokes his research into Faraday's use of construals in his critique of the "profoundly mistaken" notion "that systematic, rational thought is or can be separate from the world that it seeks to understand, manipulate or control". In collaboration with Tom Addis and others [1], Gooding builds on this work to propose a broader notion of computer science embracing 'irrational sets' that "require the use of an abductive inference system".

Of the computer-based innovations that have been developed post-Turing, the spreadsheet is perhaps the one that is most directly connected with the "modelling of states of mind" that Turing discussed in [17]. For instance, a spreadsheet can be viewed as a particularly effective way in which to represent human states of mind at the interface between the user and the computational process. Several of the characteristic themes that Hodges [10] identifies in Turing's vision of the TM also seem to be relevant to the spreadsheet concept. The spreadsheet exemplifies "the blending of the mechanical and the psychological" [10]. Through the dependency relations it characteristically maintains, the spreadsheet embodies the notion of *being determined* [10] as this is understood in two complementary ways - as in the automatic recalculation of a cell value (e.g. **profit**) from an arithmetic formula (e.g. **profit = income - expenditure**), and as in the mind of the spreadsheet user, who apprehends "profit" as indivisibly connected with "income". What is more, there is a closer correspondence between the current state of the spreadsheet and the mental state of the

spreadsheet user than in a conventional TM or procedural programming model, where the variables that are intended to record meaningful quantities (e.g. such as 'profit' and 'income') are routinely assigned intermediate values that are inconsistent with their real-world semantics. In keeping with Turing's aspirations for modelling states of mind, as characterised by Hodges in [10], this commends the spreadsheet as "a new level of description based on the idea of discrete states [such that] this level (rather than that of atoms and electrons, or indeed that of the physiology of brain tissue) [is] the correct one in which to couch the description of mental phenomena" [10].

A spreadsheet captures the human calculator's state of mind in a quite different sense from a TM. For the user of a spreadsheet, the state of interest has to do with the real-world semantics ("how will the price of petrol affect my profit?") rather than the routine computational semantics ("how is the formulae relating profit to the cost of petrol evaluated?"). The chief virtue of the spreadsheet is that it renders the mechanics of computation invisible, throwing its significance to the user into sharper relief.

Unlike a computational semantics, the real-world semantics of the spreadsheet is informal and pragmatic in character. Appreciating its state requires knowledge of the context (e.g. "to what does profit refer?"), skill in associating cells with their referents (e.g. "which cells record income and profit?") and in knowing how to carry out the interactions that disclose, maintain and probe meanings (e.g. how to change the price of petrol, how to revise the formula that define income and profit in response to changes in the tax laws, and how to carry out 'what if' experiments). The speed with which computational updates are effected, the way in which key values are disposed in the spreadsheet grid, and the level of familiarity of the user all contribute to the quality of the spreadsheet as a model of a state of mind. Turing himself discusses such issues in motivating his conception of the Turing machine: expressing concern in choosing his representations for numbers about symbols that "cannot be observed at one glance" and insisting that changes to the squares being observed "must be immediately recognisable by the computer" [17]. This is evidence that focusing exclusively on formal mathematical interpretations of TMs fails to do justice to the subtlety of his thinking. Indeed, Turing himself expresses a related concern about facile interpretations of logic when he refers [19, p15] to "a fallacy to which philosophers and mathematicians are particularly subject ... the assumption that as soon as a fact is presented to a mind all consequences of that fact spring into the mind simultaneously with it".

The goal of the EM project is to identify principles and develop tools to support a broader view of computing. Such a view takes account of roles for human agents richer than those of a 'human computer'. EM puts its primary focus on computer-based artefacts, similar in character to the *construals* introduced by Gooding in his work on the history of science, rather than 'computer programs'. An EM construal is framed with reference to three basic concepts: *observables*, *dependencies* and *agents*. These concepts have approximate counterparts in spreadsheets in - respectively - the cells, the relationships between cell values established by definitions, and the diverse modes of redefinition that are associated with state changing actions, both manual and automated, in connection with spreadsheet development and use. In keeping with the 'what if?' character of a spreadsheet, an EM construal is archetypally associated with the state of mind of a human experimenter involved (e.g.) in the kind of activity that Friedrich Steinle identifies in [16] as "exploratory experimentation" that "is driven by the elementary desire to obtain empirical regularities and to find out proper concepts and classifications by means of which those regularities can be formulated".

The relationship between the EM and TM models of states of mind is best understood by considering how exploratory activities can engineer functional machine-like entities in the world. This is illustrated by the way in which - as conceived by Gooding [8] - Faraday elaborated his construals in engineering the first prototype electric motor. EM principles for software development exploit construals in a similar way: first in enabling the exploratory sense-making activities that disclose patterns of interaction, agency and interpretation that can be reliably revisited, then in configuring the situation and exercising discretion in interaction so as to establish program-like behaviours [2]. The way in which a machine-like abstraction is here identified through a conceptual shift of viewpoint on a situation that has first been suitably engineered is something that Turing appreciated in relation to his 'discrete-state machines': "These are the machines which move by sudden jumps or clicks from one quite definite state to another. These states are sufficiently different for the possibility of confusion between them to be ignored. Strictly speaking, there are no such machines. Everything really moves continuously. But there are many kinds of machine which can profitably be thought of as being discrete-state machines. For instance in considering the switches for a lighting system it is a convenient fiction that each switch must be definitely on or definitely off. There must be intermediate positions, but for most purposes we can forget about them."

In [1], Addis, Gooding et al take inspiration from the use of construals in scientific practice but invoke the Peircean notion of abduction to arrive at a computational framework that is framed, like Turing's, in logical terms. EM gives an account of computing similar in spirit to that of Addis and Gooding in key respects but radically different in that - following William James [13] - the semantics of model-building is squarely rooted in experience. In line with James's concept of 'radical empiricism', the fundamental premise of EM is that every instance of knowing is a connection made in the present experience of an individual, and all semantic relationships must be in some way traceable to such instances. As a model for a state of mind, an EM construal is characterised by the patterns of observables, agencies, dependencies that it embodies. This is unlike a logical specification such as is expressed by abstracting variables and declaring the constraints to which they are subject. Like a spreadsheet, an EM construal represents both a current state and latent germs of change that express expectations that rely upon contextual guarantees that can never be absolute.

It is impossible to say whether Turing would have been sympathetic to such approaches to placing his fundamental contribution in a broader conceptual frame. But beyond question Turing's own style of thinking was in some respects well-matched to a pragmatic philosophical stance. And where some have made grand theoretical claims for the TM concept in relation to computer science and the mind, Turing's own outlook and working practices put the emphasis on real and topical problems, on engaging with engineering issues, and on ideas under construction, and appeal to the empiricist as well as to the logician. The remark that concludes his paper on *Computing Machinery and Intelligence* [19] testifies to the live, creative and adventurous qualities of his imagination: "We can only see a short distance ahead, but we can see plenty there that needs to be done."

Acknowledgements

I am much indebted to the many contributors to the EM research project, and especially to Steve Russ for key ideas that have motivated this paper.

References

1. Addis, T, Addis, J.T., Billinge, D., Gooding, D., Visscher, B-F. (2008). The Abductive Loop: Tracking Irrational Sets. *Foundations of Science*. 13(5), pp 5-16
2. Beynon, W.M., Boyatt, R.C., Russ, S.B. (2006) Rethinking Programming. In Proceedings IEEE Third International Conference on Information Technology: New Generations (ITNG 2006), April 10-12, 2006, Las Vegas, Nevada, USA 2006, 149-154.
3. Brooks, R. A. (1991). Intelligence without Representation. *Artificial Intelligence* 47: 139-159.
4. Brooks, R. A. (1991). "Intelligence without Reason. *Proc. IJCAI-91*. San Mateo, CA: Morgan Kaufmann. 569-595.
5. Byers, W. (2007). How Mathematicians Think: Using Ambiguity, Contradiction, and Paradox to Create Mathematics, Princeton University Press.
6. Cantwell-Smith, B. (2002). The foundations of computing. In Scheutz, M. (ed.), *Computationalism: New Directions*. Cambridge, MA: MIT Press, 23–58.
7. Feferman, S. (2006). Turing's Thesis. *Notices of the AMS.* 53(10). pp.1200-1206
8. Gooding, D. (1990). *Experiment and the Making of Meaning: Human Agency in Scientific Observation and Experiment*. Dordrecht: Kluwer.
9. Gooding, D. (2001). Experiment as an Instrument of Innovation: Experience and Embodied Thought. *Proc. 4th Int. Conf. on Cognitive Technology: Instruments of Mind.* (eds. Beynon, Nehaniv and Dautenhahn). Springer LNCS. Vol. 2117. pp.130-140
10. Hodges, A. (1988). Alan Turing and the Turing Machine. In *The universal Turing machine. A half-century survey* (ed. Rolf Herken), Oxford University Press, Oxford.
11. Hodges, A. (2004). Alan Turing: the logical and physical basis of computing. in *Proc. Alan Mathison Turing 2004: A celebration of his life and achievements*, Manchester University, June 2004. Online at http://www.bcs.org/content/conWebDoc/17127
12. Jackson, M.A. (2006). What can we expect from program verification? IEEE Computer, 39(10):53–59, October 2006.
13. James, W. (1912/1996). Essays in Radical Empiricism. (Reprinted from the original 1912 edition by Longmans, Green and Co., New York) London: Bison Books.
14. Naur, P. (1985). Intuition in Software Development. In TAPSOFT, volume 2, 60–79.
15. Post, E. (1965). Absolutely unsolvable problems and relatively undecidable propositions, in M Davis, The Undecidable, Raven Press Books.
16. Steinle, F. (1997). Entering New Fields: Exploratory Uses of Experimentation. Philosophy of Science, 64(Proceedings), pp S65-S74
17. Turing, A.M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem, Proc Lond Math Soc (2), 42, 230-265
18. Turing, A.M. (1944-5). The reform of mathematical notation (unpublished, in Collected Works)
19. Turing, A.M. (1950). Computing Machinery and Intelligence, Mind 49, 433-460
20. Turing, A.M. (1951). Programmers' Handbook for the Manchester Electronic Computer Mark II (1st edition).
21. Vincenti, W.C. (1993). What Engineers Know and How They Know It: Analytical Studies from Aeronautical History. The Johns Hopkins University Press, Baltimore.
22. Stanford Encyclopedia of Philosophy http://plato.stanford.edu/entries/turing/
23. The EM website at url: http://www.dcs.warwick.ac.uk/modelling