The Constructability of *Artificial* Intelligence (as defined by the Turing Test)

Bruce Edmonds

Centre for Policy Modelling,
Manchester Metropolitan University
http://www.cpm.mmu.ac.uk/~bruce

Abstract

The Turing Test, as originally specified, centres on the ability to perform a social role. The TT can seen as a test of an ability to enter into normal human social dynamics. In this light it seems unlikely that such an entity can be wholey designed in an 'off-line' mode, but rather a considerable period of training *in situ* would be required. The argument that since *we* can pass the TT and our cognitive processes might be implemented as a TM that, in theory, an TM that could pass the TT could be built is attacked on the grounds that not all TMs are constructable in a planned way. This observation points towards the importance of developmental processes that include random elements (e.g. evolution), but in these cases it becomes problematic to call the result artificial.

Keywords: Turing Test, Artificial Intelligence, Constructability, Evolution, Society, Culture, Computability, Symbol Grounding

1. Dynamic aspects of the Turing Test

The elegance of the Turing Test comes from the fact that it is not a requirement upon the mechanisms needed to implement intelligence but on the ability to fulfil a role. In the laguage of biology, Turing specified the niche that intelligence must be able to occupy rather than the anatomy of the organism. The role that Turing chose was a social role – whether humans could relate to it in a way that was sufficiently similar to a human intelligence that they could mistake the two.

What is unclear from Turing's 1950 paper, is the length of time that was to be given to the test. It is clearly easier to fool people if you only have to interact with them in a single period of interaction. For example it might be possible to trick someone into thinking one was an expert on chess if one only met them once at a party, but far harder to maintain the pretence if one has to interact with the same person day after day. It is something in the longer-term development of the interaction between people that indicates their mental capabilities in a more reliable way than a single period of interaction. The deeper testing of that abilities comes from the development of the interaction resulting from the new questions that arise from testing the previous responses against ones interaction with the rest of the world. The longer the period of interaction lasts and the greater the variety of contexts it can be judged against, the harder the test. To continue the party analogy, having talked about chess, one's attention might well be triggered by a chess article in next day's newspaper which, in turn, might lead to more questioning of one's acquantance.

The ability of entities to participate in a cognitive 'arms-race', where two or more entities try to 'out-think' each other seems to be an important part of intelligence. If we set a trap for a certain animal in exactly the same place and in the same manner day after day and that animal keeps getting trapped in it, then this can be taken as evidence of a lack of intelligence. On the other hand if one has to keep innovating one's trap and trapping techniques in order to catch the animal, then one would usually attribute to it some intelligence (e.g. a low cunning).

For the above reasons I will adopt a reading of the Turing Test, such that a candidate must pass muster over a reasonable period of time, punctuated by interaction with the rest of the world. To make this interpretation clear I will call this the "long-term Turing Test" (LTTT). The reason for doing this is merely to emphasise the interactive and developmental *social* aspects that are present in the test. I am emphasising the fact that the TT, as presented in Turing's paper is not merely a task that is widely accepted as requiring intelligence, so that a successful performance by an entity can cut short philosophical debate as to its adequacy. Rather that it requires the candidate entity to participate in the reflective and developmental aspects of *human* social intelligence, so that an imputation of its intelligence mirrors our imputation of each other's intelligence.

That the LTTT is a very difficult task to pass is obvious (we might ourselves fail it during periods of illness or distraction), but the source of its difficulty is not so obvious. In addition to the difficulty of implementing problem-solving, inductive, deductive and linguistic abilities, one also has to impart to a candidate a lot of background and contextual information about being human including: a credible past history, social conventions, a believable culture and even commonality in the architecture of the self. A lot of this information is not deducible from general principles but is specific to our species and our societies.

I wish to argue that it is far from certain that an *artificial* intelligence (at least as validated by the LTTT) could be deliberately constructed by us as a result of an intended plan. There are two main arguments against this position that I wish to deal with. *Firstly*, there is the contention that a strong interpretation of the Church-Turing Hypothesis (CTH) to physical processes would imply that it is theoretically possible that we could be implemented as a Turing Machine (TM), and hence could be imitated sufficiently to pass the TT. I will deal with this in section 2. *Secondly*, that we could implement a TM with basic learning processes and let it learn all the rest of the required knowledge and abilities. I will argue that such an entity would not longer be artificial in the section after (section 3). I will then conclude with a plea to reconsider the social roots of intelligence in section 4.

2. The Constructability of TMs

Many others have argued against the validity of the CTH when interpreted onto physical processes. I will not do this – my position is that there are reasons to suppose that any attempt to disprove the physical CTT are doomed (Edmonds, 1996). What I will do is argue against the inevitability of being able to construct arbitrary TMs in a *deliberate* manner. To be precise what I claim is that, whatever our procedure of TM construction is, there will be some TMs that we can't construct *or,* alternatively, that any effective procedure for TM construction will be incomplete.

The argument to show this is quite simple, it derives from the fact that the definition of a TM is *not* constructive – it is enough that a TM could exist, there is no requirement that it be *constructable*.

This can be demonstrated by considering a version of Turing's 'halting problem' (Turing, 1936). In this new version the general problem is parameterised by a number, n, to make the *limited halting problem*. This is the problem of deciding whether a TM of length[1] less than n, and input of length less than n will terminate (call this TM(n)). The definition of the limited halting problem ensures that for any particular n it is fully decidable (since it is a finite function $\{1 \ldots n\} \times \{1 \ldots n\} \to \{0, 1\}$ which could be implemented as a simple look-up table).

However there is not a general and effective method of finding the TM(n) that corresponds to a given n. Thus *what ever method* (even with clever recursion, meta-level processing, thousands of special cases, combinations of different techniques etc.) we have for constructing TMs from specifications there will be an n for which we can not *construct* TM(n), even though TM(n) is itself computable. If this were not the case we would be able to use this method to solve the full halting problem by taking the maximum of the TM and input's length finding the corresponding TM(n), and then running it for the answer. A more complete formal proof may be found in the appendix.

What this shows is that any deterministic method of program construction will have some limitations. What it does not rule out is that some method in combination with input from a random 'oracle' might succeed where the deterministic method failed. The above arguments now no longer hold, one can easily construct a program which randomly chooses a TM out of *all* the possibilities with a probability inversely proportional to the power of its length (using some suitable encoding into, say, binary) and this program could pick *any* TM. What one has lost in this transition is, of course, the assurance that the resulting TM *is* according to one's desire (WYGIWYS – what you get is what you specified). When one introduce's random elements in the construction process one has (almost always) to check that the results conform to one's specification.

However, the TT (even the LTTT) is well suited to this purpose, because it is a *post-hoc* test. It specifies nothing about the construction process. One can therefore imagine fixing some of the structure of an entity by design but developing the rest *in situ* as the result of learning or evolutionary processes with feedback in terms of the level of success at the test. Such a methodology points more towards the constructivist approaches of (Drescher, 1991, Riegler, 1992 and Vaario, 1994) rather than more traditional 'foundationalist' approaches in AI.

3. Artificiality and the Grounding of Knowledge

At the end of the previous section, I raised the possibility that an entity that embodied a mixture of designed elements and learning *in situ* (using a source of randomness), might be employed to produce an entity which could pass the LTTT. One can imagine the device undergoing a training in the ways of humans using the immersion method, i.e. left to learn and interact in the culture it has to master.

However, such a strategy, brings into question the *artificiality* of the entity that results. Although we can say we constructed the entity before it was put into training, this may be far less true of the entity *after* training. To make this clearer, imagine if we constructed 'molecule-by-molecule' a human embryo and implanted it into a woman's womb so that it developed, was born and grew up in a fashion normal to humans. The result of this process (the adult human) would certainly pass the LTTT, and we would call it intelligent, but to what extent would it be *artificial*? We know that a significant proportion of human intelligence can be attributed to the environment anyway (Neisser et al., 1996) and we also know that a human that is

1. This 'length' is the base 2 logarithm of the TM index in a suitable enumeration of machines.

not exposed to language at suitable age would almost certainly *not* pass the LTTT (Lane, 1976). Therefore the developmental process is at least critical to the resulting manifestation of human intelligence. In this case, we could not say that we had succeeded in creating a purely artificial intelligence (we would be on even weaker ground if we had not determined the construction of the original feotus but merely copied it from other cells).

The fact is, that if we evolved an entity to fit a niche (including that defined by the TT or LTTT), then is a real sense that entity's intelligence would be grounded in that niche and not as a result of our design. It is not only trivial aspects that would be need to be acquired in situ. Many crucial aspects of the entity's intelligence would have to be derived from its situation if it was to have a chance of passing the LTTT. For example: the meaning of its symbols (Harnad, 1990), its social reality (Berger, 1966) and maybe even its 'self' (Burns and Engdahl, 1998) would need to have resulted from such a social and environmental grounding. Given the flexibility of the processes and its necessary ability to alter its own learning abilities, it is not clear that any of the original structure would survive. After all, we do not call our artifacts natural just becuase they were initiated in a natural process (i.e. our brains), so why *vice versa*?

4. The Social Nature of Intelligence

All this points to a deeper consequence of the adoption of the TT as the criterion for intelligence. The TT, as specified, is far more than a way to short-cut philosophical quibbling, for it implicates the social roots of the phenomena of intelligence. This is perhaps not very surprising given that common usage of the term 'intelligence' typically occurs in a social context, indicating the likely properties of certain interactions (as in the animal trapping example above).

This is some distance from the usual conception of intelligence that prevails in the field of Artificial Intelligence, which seems overly influenced by the analogy of the machine (particularly the Turing Machine). This is a much abstracted version of the original social concept and, I would claim, a much impoverished one. Recent work has started to indicate that the social situation might be as important to the exhibition of intelligent behaviour as the physical situation (Edmonds and Dautenhahn, 1998).

This interpretation of intelligence is in contrast to others (e.g. French, 1989) who criticise the TT on the grounds that it is *only* a test for human intelligence. I am arguing that this *humanity* is an important aspect of a test for meaningful intelligence, because this intelligence is an aspect of and arises out of a social ability and the society that concerns us in a human one. Thus my position is similar to Dennett's 'intentional stance' (Dennett, 1987) in that I am characterising 'intelligence' as a characteristic that it is useful to impute onto entities because it helps us predict and understand their behaviour. My analysis of the TT goes some way to suppport this. It is for those who wish to drastically abstract from this to explain what *they* mean by intelligence – in what way their conception is useful and what domain their definition relates to (typically more abstract versions of intelligence are grounded in 'toy' problem domains).

It is nice to think that Turing's 1950 paper may come to influence academics back to considering the social roots of intelligence, and thus counter one effect of his other famous paper fouteen years earlier.

Appendix

To make the aargument about the inconstructability of some TMs more formal, one has to specify what one is intending to construct them *from*. I will take the specification to be an expression in a suitable logic, or to be more precise the index of the expression in a suitable effective enumeration of expressions. The question translates as give one's method for programming, if there is an implementation of this expression (indicated by the index of the TM is a suitable enumeration of them) can it always be obtained using this method.

In the below, *L(x)* is a predicate in a recursively axiomatisable first-order logic with equality.

If whenever we could compute the truth of *L(x)*, for any given *x* (with a program *p*), we could also compute the program *p* from the expression of *L* in the logic, then if we wanted to know whether *L* was implementable, the mere existence of such a program, *p*, would be sufficient - as we would also know that we could compute that program.

If, on the other hand, there were cases where there is a program *p* that computes *L*, but *there is no way* to compute that program *p* from *L*, then it is clear that it is, at the very least, highly misleading to say that "*L* is implementable". In such a case it may be possible to come across the program *p* by accident, but then we would still need to check that it was the correct program.

Below we show that, *given plausible limitations on our programming ability*, such cases do exist. Thus the normal characterization of computability is too weak for the purposes of the arguments in this paper. Unless there is some calculating devices more powerful than a Turing machine, in order to program *L*, *we* (aided by computers etc.) must be able to find a TM to compute it. If this is not to be the result of an unverifiable accident we must be able to somehow *effectively construct the TM that computes L.*

We can formalise the situation in the following way. *L* is a statement in some recursively axiomatized logic. So the statements in this logic can be effectively enumerated $\{L_1, L_2, \ldots\}$.

Similarly enumerate the possible TMs $\{p_1, p_2, \ldots\}$.

We then say, $(L_i, p_n) \in I$, or (L_i, p_n) an *implementation*, if

(1) $L_i(\underline{x}) \leftrightarrow p_n(\underline{x}) = 1$

and L_i is *realisable implementation*, iff

(2) $\exists n \in N ((L_i, p_n) \in RI)$

Then define $(L_i, p_n) \in EIR_m$, or (L_i, p_n) is an *effectively realisable implementation* if, in addition to condition (2),

(3) $\exists m \in N [\exists j \in N (L_i(\underline{x}) \leftrightarrow p_j(\underline{x}) = 1) \to (L_i(\underline{x}) \leftrightarrow p_{p_m(i)}(\underline{x}))]$

The program p_m represents the combined algorithm of all our ways of constructing programs from statements like L_i, in a planned, verifiable way. Of course, this may be different for different people, and at different times, but fixed for any particular person (or calculating device) at one time.

The question then becomes, given any particular p_m, encoding a systematic method of building a programs from statements in this language, *"Are there pairs* (L_i, p_n) *that are a realisable implementation but not a effectively realisable implementation?"*. In other words Is there a difference between the normal criterion of computability and such "systematically realisable computability"? The answer to this is *"Yes"*, as I now show

Theorem

$$\forall m \in N \, (\exists f, h \in N) \; (\, (L_f, p_h) \in RI - EIR_m)$$

Proof Outline

We consider a series of statements, indexed by the natural numbers, *n*, representing what I call the *limited halting problem*, i.e. *"for all* $i < n$ *(fixed) program* p_i *halts given input* $w < n$*"*. Call this $H_n(i, w)$, in contrast to the full halting predicate $H(i, w)$ (*"program* p_i *halts given input w"*). This is a finite function, hence it is computable in the sense that for each *n* there exists[1] a program q_n that decides $H_n(i, w)$. Thus for each separate *n*, $H_n(i, w)$ is theoretically reducible.

If for all n $(H_n(i, w), p_{p_m(i)}(i, w))$ is a theoretical reduction then this would allow us to also decide $H(i, w)$, since by the 's-n-m theorem' (e.g. Cutland, 1980: 81) there is a computable function, q, such that $q(x, y, z) = p_{p_m(x)}(y, z)$. Then $H(i, w)$ would be decided by $q(i+1, i, w) = 1$, and thus $H(i, w)$ would be computable, which we know is not the case (Turing 1936).

Thus for any $m \in N$ there is an $h \in N$, and a program q, $(H_h, q) \in TR$ but not $(H_h, q) \in IR_m$.

Here, you have to be careful about the indexing. What the above theorem does not say is that there is a pair that is a theoretically implementation but not an effective implementation given *any* program p_m. After all, given any particular pair one could simply add this as a special case in your plan represented by a program p_m' that would compute an index for a program to compute the first of the pair from the second. The point is that there is no *systematic* way of doing this short of adding special cases for all such cases (an infinite number of them). So if you are going to implement *all* such theoretically realisable implementations, there must be some arbitrary or non-computable element. If it is arbitrary or non-computable one can not say that it is effective. Thus at least some theoretically implementations are not systematically realisable (as in "implemented as result of a deliberate plan to do so").

1. This is not, of course, a constructive definition. One knows there *exists* such a program - it could be implemented as a simple look-up table - even if one does *not* know how to find the entries. This is the point - the criteria of computability, as normally applied, is not constructive.

Glossary of Acronyms

CTH	– Chruch-Turing Hypothesis (any text on computability, e.g. Cutland, 1980)
LTTT	– Long-Term Turing Test (defined in section 1)
TM	– Turing Machine (Turing, 1936)
TT	– Turing Test (Turing, 1950)

References

Berger, P. L., 1966, *The Social Construction of Reality*, Garden City, NY: Doubleday.

Burns, T. R., and Engdahl, E. E., 1998, The Social Construction of Consiousness, Part 2: Individual Selves, Self-awareness and Reflectivity, *Journal of Consciousness Studies*, **5**:166-184.

Cutland, 1980, *Computability*. Cambridge: CUP.

Dennett, D. C., 1987, *The Intentional Stance*, Cambridge, MA: MIT Press.

Drescher, G. L., 1991, *Made-up Minds – A Constructivist Approach to Artificial Intelligence*. Cambridge, MA: MIT Press.

Edmonds, B., 1996, 'Pragmatic Holism'. CPM Report 96-08, MMU, 1996. http://www.cpm.mmu.ac.uk/cpmrep08.html

Edmonds, B. and Dautenhahn, K., 1998, 'The Contribution of Society to the Construction of Individual Intelligence'. Workshop on Socially Situated Intelligence, SAB'98. Zurich, August 1998.

French, R. M., 1989, Subcognition and the Limits of the Turing Test.

Harnad, S., 1990, 'The symbol grounding problem'. *Physica D*, **42**:335-346.

Lane, H., 1976, *The Wild Boy of Aveyron*. Cambridge, MA: Harvard University Press.

Neisser, U. et al., 1996, Intelligence: knowns and unknowns, *American psychologist*, 51:77-101.

Riegler, A., 1992, 'Constructivist Artificial Life and Beyond'. Workshop on Autopoiesis and Perception, Dublin City University, Aug. 1992.

Turing, A. M., 1936, 'On Computable Numbers, with an application to the Entscheidungsproblem'. *Proceedings of the London Mathematical Society*, 2 **42**:230-265.

Turing A. M., 1950, 'Computing Machinery and Intelligence'. *Mind*, **59**:433-460.

Vaario, J., 1994, 'Artificial Life as Constructivist AI'. *Japanese Society of Instrument and Control Engineers*, **33**:65-71.