http://dx.doi.org/10.14236/ewic/EVA2016.42

Poetry Beyond the Turing Test

Wayne Clements
www.in-vacua.com
http://murifri.blogspot.co.uk/

Poetry conventionally is the liberated utterance of the feeling subject, "the spontaneous overflow of powerful feelings" (Wordsworth). The computer has taken poetry in several significant directions. With poetry on the page (electronic or otherwise), the computer has emulated human-authored texts. But it has also been used to create more experimental poetries. A further, better known, initiative has been to abet the departure of poetry from the page in various forms of visual, kinetic, hypertext, and hypermedia poetries. The first enterprise this paper engages with, rejects and attempts to exceed. The second it advocates. The third is not its subject.

Whitby (2002) argues that the Turing Test (the ability of the computer to deceive a human interrogator) had historical value but has since become a distraction. The utility of computers therefore lies rather in achieving what humans alone cannot do rather than what they can. Whitby argues further that The Imitation Game tests not for intelligence but for cultural similarity. In this there is a failure of design based upon an attempt to test for intelligence without quantifying what this is. There have also been a variety of Turing Tests for poetry. These continue to comprise attempts to make computer poetry pass for human authored text. Involved in this, is a frequent appeal to conventional poetics. Computer poems may emulate traditional verse forms and the poetic styles of established poets. Comparable to the familiar Turing Test, the 'Poetry Turing Test' checks for a cultural match based on accepted poetics, which in turn evokes the eloquent, speaking subject.

This paper proposes a development of poetics by the adoption of algorithmically oriented approaches to create poems based on the potential of the computer for such things as speed, or the handling of large quantities of data, or the precise control of text. Furthermore, this suggests the construal of 'the poetic' as an internally unstable and unfixed category, rather than a completed entity. This argues further for the computer as a collaborator in the further development of poetic forms. There have been a few initiatives. One of these is DIASTEXT by Jackson Mac Low. Exceptions prove rules. The combination of the algorithmic and the experimental in poetry is yet to be fully explored. To do so might add to the scope of poetry on the page.

Poetry. Poetics.Turing Test. AI. Computer. Imitation game. Experimental poetry. Formal poetry. Algorithmic.

1. INTRODUCTION

This paper defines visualisation in this sense: 'To form a mental vision, image, or picture of (something not visible or present to the sight, or of an abstraction); to make visible to the mind or imagination' (OED). Therefore, it is how poetry appears in the imagining of computing and how this might be developed that is the investigation of this paper.

The computer has had, and continues to have, a complex relation with poetry. It is possible, following Perloff, in her book 'Unoriginal Genius' (2010, p. xi), to propose that the computer has merely succeeded in assisting a multitude of kindsof animated text, so to abet the departure of poetry from the page, in various forms of multimedia. Funkhouser (2012, pp. 211–212) objects to this assessment and notes the computer's promotion of combinatory text prompted by user intervention. However, it is 'poetry on the page' (electronic or otherwise), or 'a "normal" print poem' (Perloff, 2010, p. xi) with which this paper mainly is concerned. However, it must be acknowledged, the distinction between writing on some form of electronic page and poetry that takes flight from it may in practice be hard to maintain, as digital writing is extremely protean and diverse. Nevertheless, despite this caution, it is forms of 'normal' print poems with which we are concerned.

In 'the print poem', there are two leading developments that this paper explores. One is

where the computer has emulated human-authored texts, in a 'Poetry Turing Test'. The other use of the computer has augmented and added to poetry writing and to poetic form. It is the distinction between these two developments that is the chief focus of this paper.

Discussion of the second of these initiatives is perhaps not so well developed. It is possible to theorise why this might be. Poetry, conventionally, is the liberated utterance of the feeling subject, "the spontaneous overflow of powerful feelings" (Wordsworth). It is literature's commitment to the rhetoric of the feeling subject that is today the primary reason that poets largely, it may be argued, continue to use computers merely as augmented typing/writing machines.

Programmers may, in contrast, be interested in the application of the computer to poetry. But their interest in doing this is to enable the computer's poetic efforts to pass for human authored poetry. This venture prompts that the computer write poetry that is not experimental. Rather, 'Turing Test Poetry' is dependent on established models if it is to pass the test.

'Turing Test Poems' are generally written by programmers with a side-line in poetry. Poets are not really interested in a fundamentally computational problematic. Poets may, however, if their work is rule-based, be interested in the mechanisation of poetic practises that may have already existed off-line. Here follow a few examples, by no means a complete list.

There is, for instance, Brion Gysin's 'I am that I am', a very early (1959) permutation poem completed in collaboration with the Cambridge computer programmer Ian Summerville. Nanni Balestrini's 'Tape Mark 1' (1961) is an instance of a computer used to shuffle lines of source texts to create new combinations. As such, it is essentially a computer version of Tzara's 'To Create a Dadaist Poem' of 1920. Balestrini collaborated with IBM computer scientists to create the poem (Amaral, no date p. 42). Another is John Cage's Mesolist, of 1984, a program written by Jim Rosenbergfor creating *mesostics*, where a word is spelt vertically in the middle of text, similar to an acrostic. This is a mechanisation of an existing Cagean compositional method (Funkhouser, 2007, p.64). A further example of text manipulation is DIASTEXT by Jackson Mac Low, to which we will return later.

In the early days, artists and poets quite often cooperated with programmers in a form of officially sanctioned mainframe computer literature. Early computer experiments were somewhat hampered by the difficulties in gaining access to computers and the further problem of a lack of programming languages that handled text strings readily. This helps explain why algorithmically oriented authors often did not progress far in writing their own computer programs. (An instance is Jackson Mac Low. It seems that despite collaborative work with programmers, Mac Low never sufficiently mastered programming. Mac Low's son says he took courses in Fortran, for instance, but says in the conclusion to an essay about his father: 'He made at least one further attempt to learn to program, this time in C, but in the end he relied primarily on already written software, albeit occasionally pushed to its limits'. See Mac Low (2012, p. 306.)

But currently, institutional barriers do not stand in the way of access to computers. It is tempting to infer that the difficulty of deploying programming languages may now be the significant impediment. Computer languages, designed to handle text, such as PERL, are powerful but hard to learn. Yet the contention of this paper is that the seeming disinclination of poets to try to acquire a command of programming is dependent on their prior conception of what poetry is, and on their established poetic practice (not technical problems alone).

In the face of an early encounter with computers in literature, the poet Gregory Corso (Bailey, 1974, p.292) divided up poetry into 'soul poetry' and 'word poetry'. (His) poetry was to be 'written from the soul and not from the dictionary'. The use of computers in poetry continues to raise this apparent conflict. In truth, the polarised way in which Corso posed these contrasts does not really represent what is in fact a complex and much imbricated state of affairs. Corso's contradiction between a concentration on feeling instead of on text echoes a similar contrast inscribed into the history of modernism in literature and described by Perloff. Perloff (pp. 1–3) refers to an early review of Eliot's 'The Wasteland'. The substance of the reviewer's reaction to Eliot was that the work was not written from the soul ('too reserved to expose in public the impressions stamped on his own soul', pp. 1-2) but from quotation and citation of existing literature. The avoidance of, or repulsion from, the computer as a compositional tool thus risks a recursion to pre-modern typifications of the poetic subject: the privileging of the poetic self over an engagement with words.

Although many poets would resist being forced to choose between *words* and *feeling*, it is possible to contend that poets do not learn computing, at least in part, because the poetry they write has no use for the computer (other than as a glorified tablet for inscription). They do not write, in the main, with rules and instructions. If they were to write algorithmically, the utility of the computer would become more apparent and might more strongly

motivate acquisition of computer programming skills. This would be to write what Epstein and Schneider man (1977, p. 1) referred to a long time ago as 'formal poetry', that is, 'any poem that has an algorithmic or programmable structure'. Most poets do not write this way and thus do not take the next step and progress to computer programming. Thus, despite the efforts of the OULIPO and their computer wings, the ALAMO and the TEANO, formal poetry continues to be less developed (see Wolff, 2007; or Mathews and Brotchie, 2011).

2. OVERVIEW

This paper advocates a comparable development in poetry that uses computers to that in computing more generally. It proposes an expansion of poetics by the adoption of algorithmically oriented approaches to create text based on the sort of things the computer is good at, such as speed and the manipulation of large quantities of data, and the computer's potential for manipulating text in unusual ways. We will return to these below. But this suggests the construal of 'the poetic' as an internally unstable and unfixed category rather than a completed entity. This argues for the computer as a collaborator in the further development of poetic forms. This is to favour sorts of 'computer assisted poetry' (to use a familiar typology, derived from Bailey, 1974). Such an approach remains largely 'poetry in potential' (to borrow a word from the OULIPO). Such advocacy goes beyond the revisiting of modernist practices refashioned for the Internet. Of the latter, Goldsmith (2011), for instance, is interested in the possibilities for appropriation and cut-up, surely two of the most established compositional methods associated with modernism.

3. POETRY

The further development of 'formal poetry', as a type of computer assisted poetry, contrasts with 'computer poetry', where, in the latter, the computer is encouraged to generate the poem with the minimum intervention of a human. In practice this may be a hard distinction to draw, one of tendency rather than of absolute kind. Automated poems on the WWW are one form of 'computer poetry'. In this genre, texts may be produced with minimal contribution from the user.

An example might be an automated version of Queneau's 'Cent mille milliard de poèmes', such as may be found on Florian Cramer's 'permutations' website (http://permutations.pleintekst.nl/). Others may be found on the author's www.in-vacua.com website. Another form of computer poetry, however, is that which this paper refers to as 'Turing Test Poetry'.

With Turing Test Poetry another danger arises. Instead of the augmentation of poetic practice, in the form of 'computer assisted poetry', by digital computers the premature replacement of the poet by pure computer poetry is conjectured. This, in its most unalloyed form, is posed in terms of attempting successful Poetry Turing Tests. But as we shall see, it is questionable whether the poems in these tests were really written by the computer alone. Real or otherwise, successful or not, the use of poetry in AI research has not been a progressive step, either for computing or for poetry, it may be argued.

Whitby (2002) argues, in general, that the Turing Test (the ability of the computer to deceive a human interrogator) had historical value but has since become a distraction. The utility, for Whitby, of the computer therefore lies assisting in achieving what humans alone cannot do rather than what they can, 'to enable a human operator to achieve much more than he could without it' (Whitby, p. 59)

Whitby argues that The Imitation Game tests not for intelligence but for cultural similarity. In this there is a failure of design based upon an attempt to test for intelligence without quantifying what this is. This, cultural similarity, it is argued in this paper, is the product of much research dedicated to the Turing Test in so far as it applies to poetics. However, Turing's discussion of The Imitation Game, in his 1950 paper 'Computing Machinery and Intelligence', in fact proposes replacing the question, 'Can machines think?' with a different question.

The experiment is carefully defined. Turing (1950) writes:

> 'The new form of the problem can be described in terms of a game which we call the "imitation game." It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman.'

In this closely defined experiment the original question can be replaced. 'We now ask the question, "What will happen when a machine takes the part of A in this game?" Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, "Can machines think?"' (ibid.).

There have been, it may be observed, variants of a Turing Test for poetry. This continues to comprise an attempt to make computer poetry pass for human authored text. Involved in this, there is a

frequent appeal to a kind of conventional poetics. Computer poems may emulate traditional verse forms and the poetic styles of established poets. Comparable to the familiar Turing Test, the 'Poetry Turing Test' may be argued to test for a cultural match based on a notion of already accepted poetics.

Accountability for the imitation of established poetic models by computers probably rests, it would seem, with Turing himself. In 'Computing Machinery and Intelligence' he proposed a 'sonnet machine' as a test of The Imitation Game. In a London Times interview in 1949 Turing elaborated further. '... I do not see why it [the computer] should not enter any one of the fields normally covered by the human intellect, and eventually compete on equal terms.' He went on, 'I do not think you can even draw the line about sonnets, though the comparison is perhaps a little bit unfair because a sonnet written by a machine will be better appreciated by another machine' (quoted in Hodges, p. 406).

Here Turing raises the question of a different set of critical standards for judging computer sonnets. Implied in this is that computer sonnets are a different order of poetic composition to human authored. However, those tempted by Turing's invitation to create poetry-writing computer programs, largely model their work on existing poetic examples. They do not on the whole view the computer as a method of revolutionising poetic form. The reason for this is not very hard to suppose, as mentioned above. It is true that Turing Test Poetry is usually created in collaboration with the computer by authors who are first and foremost computer scientists. They are only secondarily poets. They are not, in the first instance poets, who happen to be able to program. However, perhaps more significantly, innovation is ruled out by the structure of the Turing Test. In poetry, this tests for conformity to established poetic models: *does it look like a poem?* Poems that refuse to conform to accepted models might be more likely to be doubted.

The acceptance of the challenge to create plausible poems generated by computer differs from Turing's own treatment of the idea of a 'sonnet machine' in his 1950 article. While the computer in The Imitation Game willingly adds up numbers and makes moves in chess, it refuses to write a sonnet: 'Count me out on this one. I never could write poetry.' (Turing, 1950, p. 2) The implication is that the *inability* to write poetry to order approximates more closely to human behaviour than the ability. Knocking off a quick sonnet is an unusual accomplishment – and one that might be suspected.

More recent versions of a Poetry Turing Test attempt to circumvent this problem of the difficulty of poetic composition done to order by altering the parameters of the test. Poems are completed in advance. Then they are made available for evaluation by human readers, who may not know their authorship.

Claims, typically, may be, high. Professor Philip Parker states of his work in generating texts: 'A computer works very well with rules and the most obvious way is poetry'. He describes his own Turing test in poetry: We did a blind test between a Shakespearean sonnet and one that the computer had written. A majority of people surveyed preferred ours'. He points out reasonably that this result should be treated with caution: 'That's not to say it was better, Shakespeare is a genius, but it was what people preferred' (Hudson, 2012, no page numbering).So it is not very clear that the poems passed as human authored, but the inference may be taken that they did.

Another example is Swift-speare, a mobile 'phone app. It was reported in the online magazine TechCrunch with the headline: 'Computer Algorithm Generates Poetry As Good As Shakespeare's' (Brooks, 2014, no page numbering). The article continued, 'J. Nathan Matias used the popular machine-learning Android app. Swiftkey in combination with some of Shakespeare's famous words to produce sonnets that rival the master himself.' The article included a Shakespearian sonnet that not only kept to a pretty steady iambic pentameter, but also rhymed. This would be a considerable achievement for a machine alone. The article included the caveat, 'Machine-learning still requires a human element, which means it won't replace human poets anytime soon' (ibid.). But the extent of human intervention was not made plain. The impression remained that the app. had done most of the compositional work. But this was not so. When asked about this Matias replied, 'I was writing in collaboration with *suggestions* from the system, rather than automatically generating a text with no human supervision' (Matias, 2016, Private Communication). A version of a Markov chain algorithm modelled likely strings of words based on a 'reading' of Shakespeare's writings. Appropriate snippets were then pieced together by the user to make the finished poem:

> When I in dreams behold thy fairest shade
> Whose shade in dreams doth wake the sleeping morn
> The daytime shadow of my love betray'd
> Lends hideous night to dreaming's faded form
>
> From,Matias (in Brooks, 2014) Untitled.

In this method, Markov chain algorithms, we encounter one of the main ways of generating texts using a computer. Markov chains were employed for instance in the program that created the quite well known character Mark V Shaney, an early (1980s) instance of a computer entity that passed for the creator of human authored text. (See Kernighan and Pike, 1999, p.84.) Markov algorithms have been utilised, with variants, by programmers ever since. The method for producing text by Markov chain algorithms is beyond the scope of this paper to explain. This is also true of another popular method of generating texts, recursive transition networks. This is a method that aims for greater apparent coherency by its use of a grammar file of text through which the computer navigates. The use of recursive transition networks is described in Hoftstadter (1999) and in Bulhak (1996). There are also text shuffles and 'slot systems' (where words are dropped into templates). The technical aspects, on the computer, of these too preclude a discussion.

Similar methods to that of Swift-speare are used by Kurzweil's Cybernetic Poet ('RKCP'). 'Version 1 of RKCP was based on simple Markov-like models', according to Kurzweil's (2000, no page numbering) description. While, 'RKCP Version 2 is significantly more sophisticated in its modelling techniques, uses more complex Markov models'. 'A (Kind of) Turing Test' (Kurzweil, 1999, no page numbering) describes an experiment that Ray Kurzweil conducted in the late 1980s in which he gave poems to 16 human (both adult and child) judges. In this test, he gave each judge 28 poems, some of which were written by RKCP and some of which were written by human poets, the same poets used to create the poet personalities that RKCP used. The judges were asked if they could tell which poems were written by a human poet and which were written by the computer. 'While the adult judges scored somewhat better than chance (63 percent), their answers were far from perfect. The computer poet was able to trick the human judges much of the time'.

> is beauty itself
> that they were walking there. All along the new
> world naked,
> cold, familiar wind–
>
> Kurzweil (1999) Untitled.

This according to Kurzweil (ibid.) was 'written by the Kurzweil Cybernetic Poet after reading poems by William Carlos Williams'.

People may not be very competent judges of authorship. They may lack the experience to tell what human-authored text is. Be this as it may, a further point here is that the Cybernetic Poet uses Markov processes to model new texts based on what it has 'read' and that this results in poems that tend to echo existing writers' manners.

But not all computers, in the field of poetry, that attempt the Turing Test, necessarily build a poem based on a reading of existing, human authored poems. Zackary Scholl describes one such in, 'The Poem That Passed the Turing Test' (Scholl, 2015, no page numbering). The story goes that in 2011 the editors of a student-run literary journal selected a short poem called 'For the Bristlecone Snag' for publication in its Fall edition. This is not writing to order, of course, as in Turing's original refused gambit in The Imitation Game scenario, we may recall. There are other demurs, such as we cannot be sure what the editorial view of the poem was, other than that it was accepted for publication. But the poem is interesting in that it is generated recursively using a grammar file: '…it recursively generates the larger segments (phrases, sentences) and then fills in the smaller segments (nouns, verbs, adjectives)' (Scholl, 2016, Private Communication). As such, the poems produced are less dependent on poem-models and are less derivative in style perhaps than those that use Markov processes. The following is an excerpt from a poem to be found on Scholl's website:

> I conduct myself in a windy manner because I
> am
> drunk and enchanted in this field.
> The oxygen around my head is rabid
> and filled with orange light
> like a equinoctial tiger to its flesh.
> My heart moves violently
> on this neon ship.
>
> From,Scholl (2015) Orange Light.

It does suffer, however, from what Bailey (1974) identified as a flaw. It reflects what the 'creator thinks a poem should look like' (p. 286). In this case we have the intoxicated voice (*drunk*, *enchanted*) of the poet in the grip of powerful emotions (*rabid*, *violently*). As such, the poetics are those of cliché. This may easily be no criticism of the creator, Scholl, if deployed knowingly.

But this may serve to lend substance to the argument posed by the poet and programmer Charles Hartman,'… the Imitation Game again. The trap for poetry is that the more accurately the computer mimics human language, the more ordinary it becomes. In fact, the ordinariness is how we measure the accuracy of imitation' (Hartman, 1996, p. 94).

Examples by artists and poets, rather than programmers, avoid ordinariness in favour of what Hartman refers to as an 'unstable balance of the plain and strange' (ibid. p. 95). In these initiatives the emphasis is not on imitation but innovation.

One of the better known computerised text writing techniques is probably DIASTEXT (mentioned above). It is a technique first created 'offline' by Jackson Mac Low, but later programmed for computer by Hartman. The technique is in essence to have a key word and proceed through a text looking for words with each letter in the same position (first letter in the first position, second in the second, and so on):

> **E**lysium is as far as to
> I**M**pregnable of Eye-
> Th**I**s was in the White of the Year-
> Neg**L**ected son of Genius
> Rudd**Y** as that coeval Apple
>
> From, Quatorzains from & for Emily Dickinson
> (Mac Low, 2008, p. 175).

Despite its classical reference and mention of Genius (probably ironic rather than conventional in this case), the stanza avoids predictability, in part by the fragmentation of syntax, in part by the penetration of the text by the vertical graphic of the buried term '**EMILY**'.

This technique is therefore a text generation method, producing a variant text from abase text. DIASTEXT exists along with some other procedures that perform generative processes on pre-existing, source texts. Computers may therefore contribute, for instance, amounts of unexpected strangeness through their manipulation textual databases. This is not, however, the main trajectory that has been taken when it comes to poetry and computing. The main development has not been to augment what we referred to above as 'normal', albeit innovative, poetry (poetry on the page). Nor has the dominant approach been to take part in the sub-genre of Turing Test Poetry. Instead, it has been the formation of new works of visual, kinetic, hypertext, and hypermedia poetries that have dominated. This paper does not oppose these latter developments, but argues for the harnessing of the computer to the creation process of poetry on the page.

To merit this, what we need is new algorithms that might benefit from the computer's capabilities. If we have these, and if they are productive of novel poetries, they might be programmed using a computer. Exceptions prove rules. The combination of the algorithmic and the experimental in poetry is yet to be fully developed. To do so would also occasion the augmentation of the poetic forms by means of computer exponentiation.

4. FINALE

The following poem was generated by permutating a passage (in translation) of Alain Robbe-Grillet's novel 'Jealousy'. The program used is named 'PAGE1' (there are currently three: PAGE, 'Poetic Algorithmic Generative Executable'). It is written in PERL. Its method is, in essence, to work backwards through a text in overlapping two word/phrase blocks.

DEPARTURE

it is it
 it is doubtless it
is doubtless the same
 is doubtless the same poem
doubtless the same poem
 continuing the same
poem continuing if poem continuing
 if the themes –
continuing–
 if the themes sometimes –
if the themes sometimes blur
 the themes sometimes blur
they sometimes blur
 they only blur
they only recur
 they only recur somewhat later
only recur somewhat later
 all recur somewhat later
all the more somewhat later
 all the more clearly
all the more clearly
 virtually the more clearly
virtually identical
 clearly virtually identical
yet these virtually identical –
 yet these repetitions –
identical –
 yet these repetitions
these yet these
 repetitions, these tiny repetitions
these tiny variations
 these tiny variations
halts, tiny variations, halts, regressions
 variations, halts, regressions can –
halts, regressions can give rise
 regressions can give rise to modifications
can give rise to modifications
 though barely give rise to modifications
though barely perceptible
 to modifications
though barely perceptible
 eventual, though barely perceptible
eventual, moving, perceptible, eventual
 moving quite far
eventual, moving quite far from the point–
 moving quite far from the point of departure
quite far from the point of departure
 of departure from the point of departure

Wayne Clements

5. REFERENCES

Amaral, C. (no date) Edições Críticas Digitais: O Caso da Literatura Electrónica de Pedro Barbosa. https://repositorioaberto.up.pt/bitstream/10216/79543/2/117861.pdf (retrieved 5 March 2016).

Bailey, R. W. (1974) Computer-assisted poetry: the writing machine is for everybody in, Computers and the humanities. In J. L. Mitchell (ed.) Edinburgh University Press, Edinburgh, 288–296.

Brooks, R. (2014) Computer Algorithm Generates Poetry As Good As Shakespeare. http://www.psfk.com/2014/01/shakespeare-machine-learning-poetry-app.html (retrieved 5 March 2016).

Bulhak, A. C. (1996) On the Simulation of Postmodernism and Mental Debility using Recursive Transition Networks. http://www.csse.monash.edu.au/cgi-bin/pub_search?104+1996+bulhak+Postmodernism (retrieved 2 August 2004).

Epstein, G. and Schneiderman, B. (1977) Technical Report No. 62. Formal and Computer Poetry. ftp://ftp.cs.indiana.edu/pub/techreports/TR62.pdf (retrieved 5 March 2016).

Funkhouser, C. T. (2007) Prehistoric Digital Poetry, an archaeology of forms, 1959–1995, University of Alabama Press, Tuscaloosa.

Funkhouser, C. T. (2012) New Directions in Digital Poetry, Continuum, New York.

Goldsmith, K. (2011) Uncreative Writing, Columbia University Press, New York.

Hartman, C. O. (1996) Virtual Muse: Experiments in Computer Poetry. University Press of New England, Hanover, NH.

Hodges, A. (1983) Alan Turing: The Enigma of Intelligence. Unwin, London.

Hoftstadter, D. R. (1999) Gödel, Escher, Bach: An Eternal Golden Braid, Penguin, Harmondsworth, Middlesex.

Hudson, A. (2012) Man or machine – can robots really write novels? BBC News, BBC, UK. http://news.bbc.co.uk/1/hi/programmes/click_online/9764416.stm (retrieved 5 March 2016).

Kernighan, B. W. and Pike, R. (1999) The Practice of Programming, Addison-Wesley, Reading, MA.

Kurzweil, R. (1999) A Kind of Turing Test http://www.kurzweilcyberart.com/poetry/rkcp_akindofturingtest.php. (retrieved 5 March 2016).

Kurzweil, R. (2000) History of RKCP http://www.kurzweilcyberart.com/poetry/rkcp_historyofrkcp.php. (retrieved 5 March 2016).

Mac Low, J. (2008) Thing of Beauty, University of California Press, Berkeley, CA.

Mac Low, M. (2012) The Role of the Machine in the Experiment of Egoless Poetry: Jackson Mac Low and the Programmable Film Reader. In H. B. Higgins and D. Kahn (eds.), Mainframe Experimentalism: Early computing and the foundations of the digital arts, University of California Press, Berkeley, CA, pp. 298–311.

Matias, J. N. (2016) Private Communication.

Perloff, M. (2010) Unoriginal Genius: Poetry by Other Means In The New Century. University of Chicago Press, Chicago.

Scholl, Z. (2015) The Poem That Passed the Turing Test. https://rpiai.wordpress.com/2015/01/24/turing-test-passed-using-computer-generated-poetry/ (retrieved 5 March 2016).

Scholl, Z. (2016) Private Communication.

Turing, A. (1950) Computing Machinery and Intelligence. Mind, 49, pp. 433–460. http://www.csee.umbc.edu/courses/471/papers/turing.pdf (retrieved 5 March 2016).

Whitby, B. (2002) The Turing Test: AI's Biggest Blind Alley? In P. Millican and A. Clark (eds.), Machines and Thought. The legacy of Alan Turing. Volume. 1, Oxford, University Press, pp. 53–63.

Wolff, M. (2007) Reading Potential: The Oulipo and the Meaning of Algorithms. http://www.digitalhumanities.org/dhq/vol/1/1/000005/000005.html (retrieved 5 March 2016).