International Journal of Theoretical Physics, Vol. 44, No. 11, November 2005 (© 2005)
DOI: 10.1007/s10773-005-8984-0

Computational Power of Infinite Quantum Parallelism

Martin Ziegler[1,*]

Received October 31, 2004; accepted June 30, 2005

Recent works have independently suggested that quantum mechanics might permit procedures that fundamentally transcend the power of Turing Machines as well as of 'standard' Quantum Computers. These approaches rely on and indicate that quantum mechanics seems to support some infinite variant of classical parallel computing.

We compare this new one with other attempts towards hypercomputation by separating (1) its computing capabilities from (2) realizability issues. The first are shown to coincide with recursive enumerability; the second are considered in analogy to 'existence' in mathematical logic.

KEY WORDS: Hypercomputation; quantum mechanics; recursion theory; infinite parallelism.

PACS (2003): 03.67.

1. COMPUTABILITY

In 1936, Alan M. Turing gave an example of a well-founded (and thus mathematically solvable) problem, which he showed to admit no computable solution. More precisely in Turing (1936) he introduced what is now called the Turing Machine (TM) as an idealization ('*model*') of a digital computer and revealed that it was capable of solving a vast variety of practical problems like, for instance, deciding whether a given integer $x \in \mathbb{N}$ is prime, that is, belongs to $L := \{p \in \mathbb{N} : p \text{ prime}\} \subseteq \mathbb{N}$ or not.

Definition 1.1. A computational *problem* is a subset $L \subseteq \mathbb{N}$. It is *decided* by TM M if, upon input of any $x \in \mathbb{N}$,

- M eventually outputs "0" (*rejects*) and halts in case $x \notin L$.
- M eventually outputs "1" (*accepts*) and halts in case $x \in L$.

Having thus indicated the fundamental power of this machine, Turing then proceeded to exhibit its limitation by formally proving that the Halting Problem

[1] University of Paderborn, Paderborn 33095, Germany; e-mail: ziegler@upb.de.
*Supported by DFG project Zi1009/1-1.

H—the question whether a given TM M eventually halts or rather continues executing indefinitely—cannot be decided by any TM M_0. Notice that, according to Definition 1.1., this hypothetical M_0 is required to always give the correct answer and to terminate. More precisely, the difficulty inherent to the Halting Problem consists in telling within finite time whether M does *not* halt; for, simply simulating M step-by-step, M_0 can easily identify the case when M *does* terminate.

Turing's result initiated the flourishing field of Computability or Recursion Theory (Odifreddi, 1989). Its goal is to distinguish computable from uncomputable problems and to classify the latter according to their degree of uncomputability (Soare, 1987). For example, the following celebrated result of Matiyasevich has settled Hilbert's Tenth Problem in the negative by proving it equivalent to H:

Theorem 1.2. (Matiyasevich (1970)) *On the one hand, a given description of a Diophantine Equation E (like Fermat's famous "$a^n + b^n = c^n$" for $a, b, c, n \in \mathbb{N}$) can computably be transformed into a TM M such that it holds: M terminates iff E admits an integral solution.*

On the other hand, a given TM M can computably be converted into the description of a Diophantine Equation E in such a way that, again, M halts iff E admits an integral solution.

Apart from the TM, many further sensible notions of computability have been proposed: e.g., μ-Recursion (which gave the field its name), Herbrand–Gödel-Computability (which led to the programming language `Prolog`), and λ-Calculus (which stipulated `Lisp`). But they were all shown equivalent to the TM by Church, Kleene, Post, Turing, and others; cf. e.g., Odifreddi (1989, Chapter I) or Atallah (1999, Section 26.3+4). Even nowadays PCs are still basically just TMs (the processor corresponds to the finite-state control and the infinite tape models the computer's finite but adaptively extendable storage) although very fast ones; recall that we are dealing with problems that cannot be solved computationally at all, neither quickly nor slowly. For the very same reason, (at least 'standard') Quantum Computers are still no more powerful than an ordinary TM (Gruska, 1999, p. 3, footnote 1).

1.1. Church-Turing Hypothesis

It should be stressed that both Halting and Hilbert's Tenth Problem are desirable to be solved for very practical reasons. The first for instance arises in automated software verification; indeed, correctness of a program includes its termination which, by the above considerations, cannot be checked algorithmically.

Similarly, a hypothetical algorithm for deciding feasibility of Diophantine Equations could be applied to computer-proving not only Fermat's Last Theorem but also to settle many other still open questions for example in number theory.

Observing that Turing (1936) and Matiyasevich (1970) ruled out the possibility of a *Turing Machine* to decide either of these problems, people have since long tried to devise other computing devices exceeding its power. However, the perpetual failure to do so plus the aforementioned results of a TM being able to simulate many other notions of computability have eventually led to what has become known as the Church–Turing Hypothesis:

Anything that can be computed in practice, is also computable by a TM.

We emphasize that, due to its informal nature, this hypothesis cannot be proven formally. Arguments in its favor usually point out that computation is a physical process which, by mathematically describing the physical laws it is governed by, can be simulated by a TM up to arbitrary finite numerical precision; and infinite accuracy were required only for 'chaotic' processes that are too sensitive to perturbations than being harnessable for practical computation anyway.

However, it has later been pointed out that certain theories of quantum gravitation might actually *not* admit a simulation by a TM (Geroch *et al.*, 1986, p. 546); furthermore even classical mechanics seems to conceivably provide for processes whose simulation requires infinite precision during *intermediate* times only, whereas the resulting behavior is asymptotically stable and thus suited well to realize a non-Turing form of physical computation (Yao, 2003). Moreover, the laws of nature we know so far seem to restrict (if at all) the performance but not the fundamental capabilities of computation (Bennett, 1985).

In fact, neither Church nor Turing themselves have put forward a claim as bold as the way 'their' hypothesis is often (mis-)interpreted (Copeland, 1997). Instead, the literature contains and discusses a rich variety of related hypotheses (Ord, 2002, Section 2.2).

1.2. Hypercomputation

Anyway, the question remains open whether there might exist a computing device more powerful than the TM or not. To get an idea how such a *Hyper*computer might look, theoreticians have started considering super-TMs and their respective fundamental computing capabilities. This established the flourishing field of research called 'Hypercomputation' (Copeland *et al.*, 1999) that entire volumes of significant journals have been dedicated to (Calude *et al.*, 2002a; Burgin *et al.*, 2004). Devising such a formal model (i.e., an idealized abstraction)

of a hypercomputer proceeds in many cases less by adding extensions to than by removing restrictions from a TM.

Observation 1.3. *The (decidability by a) TM is characterized by*

(a) a finite control (the 'program,' so to speak);
(b) an initially blank, countable supply of memory cells
(c) storing a finite amount of information each (e.g., a bit or an integer);
(d) finite running time;
(e) possibly finite parallelism (as, e.g., for a nondeterministic *TM).*

Regardless of the details of its precise definition, these finiteness conditions directly imply that there is an at most countable number of different TMs; whereas computational problems according to Definition 1.1. exist of cardinality of the continuum. Thus, most of them are undecidable.

Conditions (a)–(e) underlie the mourned limitations of the classical TM, and dropping one or more of them leads to several well-known models of hypercomputation; see, e.g., Ord (2002, Section 3) or Copeland (2002, Section 2). *Oracle* machines for instance, subject of Turing's dissertation in Princeton (Turing, 1939) and now core of Recursion Theory (Odifreddi, 1989; Soare, 1987), correspond to TMs with initial memory inscription, that is, they remove Condition (b); Blum, Shub, and Smale's $\mathbb{R}$-Machine (Blum *et al.*, 1989, in particular Section 1, Example 6) abolishes Condition (c) by allowing each cell to store a real number; while *Infinite Time* Machines due to Hamkins *et al.* (2000) lift Condition (d).

The proposal, consideration, and investigation of such enhanced abstract models of computation and their respective computational powers by logicians and theoretical computer scientists has proven particular seminal regarding related contributions from Theoretical Physics on their realizability. For example, Beggs *et al.* (2004) has indicated that a physical system breaking Condition (a) might actually exist[2]; while Hogarth (1992); Etesi *et al.* (2002); Shagrir *et al.* (2003) pointed out that in General Relativity there might exist[2] space–time geometries allowing one to watch within finite time a computer execute an infinite number of steps and thus to lift Condition (d).

1.3. Quantum Mechanical Hypercomputation

Recently, several new approaches have been suggested for solving either the Halting Problem (Adamyan *et al.*, 2004; Calude *et al.*, 2001, 2002b) or Hilbert's Tenth Problem (Kieu, 2003a,b). They exploit quantum mechanics and thus form a nice counterpart to previous approaches based on General Relativity (Etesi *et al.*, 2002; Hogarth, 1992; Shagrir *et al.*, 2003) as the other pillar of

[2] Refer to Remark 1.4.

non-classical physics. Recalling that 'standard' Quantum Computing does not exceed Turing's Barrier, these approaches must be non-standard in some sense which closer inspection reveals to be infinite parallelism:

- "*Our quantum algorithm is based on [. . .] our ability to implement physically certain Hamiltonians having infinite numbers of energy levels*" (Kieu, 2003a, top of Section 6.3);
- "*The key ingredients are the availability of a countably infinite number of Fock states, the ability to construct/simulate a suitable Hamiltonian*" (Kieu, 2003b, end of Section 4);
- "*The new ingredients built in our 'device' include the use of an infinite superposition (in an infinite-dimensional Hilbert space) which creates an 'infinite type of quantum parallelism'* " (Calude *et al*., 2002b, p. 123, Section 5).

It thus seems that quantum mechanics allows to drop Condition (e) from Observation 1.3 and so to provide a new promising approach to the existence[2] of hypercomputers—an approach *not* included in Ord's classification (Ord, 2002, Section 3). The present work describes in Section 2. the theoretical consequences from lifting Condition (e), that is the computing power of infinite parallelism.

We conclude this section with an already announced remark on the notion of *existence*.

Remark 1.4. The question whether a physical device with certain properties *exists* bears logical similarity to the question whether a mathematical object with certain properties exists. In the latter case for a proof, only very few (namely constructive or intuitionistic) mathematicians will

(A) insist that one actually *constructs* this object

whereas most are satisfied for instance with

(B) an indirect argument showing that its *non*-existence leads to a contradiction.

In fact a majority of contemporary mathematicians will even take it for granted if

(C) the object's *existence* does *not* lead to a contradiction.

For example, the claim "*For every vector space there* exists *a basis*" is of kind (C) as are many principles in Functional Analysis: Each of them is equivalent to the Axiom of Choice (Blass, 1984) and thus does not lead to a contradiction to conservative set theory (C) but cannot be deduced from it (B) as has first been proven by K. Gödel in Gödel (1940) and later strengthened by P. J. Cohen.

Similarly, the *existence* of a physical object can be proven in a strong way (A) by actually constructing it. But in most cases, showing it (C) consistent with physical laws is accepted as well. Observe that this is how both positrons and black holes came into 'existence': As solutions of (and thus consistent with)

Dirac's Equation and Einstein's General Relativity, respectively; only later have new experimental observations upgraded our conception of their *existence*.

2. INFINITE PARALLEL COMPUTING

The prospering field of Parallel Computing knows and has agreed upon a small collection of models as theoretical abstractions for devising and analyzing new algorithms for various actual parallel machines (Atallah, 1999, Sections 45.2 and 47.2). Of course with respect to their principal power, that is computability rather than complexity, they are all equivalent to the TM.

However, when talking about *infinite* parallelism, seemingly no such agreement has been established, cf. e.g., Eberbach *et al*. (2003, p. 284); and in fact no equivalence, either, as it will turn out. For instance, of what kind are the countably infinitely many individual computers that are to operate concurrently—TMs or finite automata? In the first case, do they all execute the same program? When is the result to be read off? The answers to these questions fundamentally affect the capabilities of the resulting system.

2.1. Infinite Cellular Automata

Consider parallelism in an infinite cellular automaton in the plane. More specifically, we refer to Conway's famous Game of Life (Berlekamp *et al*., 2004, Chap. 25) where in each step, any cell's successor state concurrently is determined by its present state as well as those of its eight adjacent ones' as follows (cf. Fig. 1):

- A `dead` cell with exactly three neighbors `alive` becomes `alive`, too; otherwise it remains `dead`.
- A living cell with two or three neighbors `alive` stays `alive`; otherwise (0,1,4 . . . 8 living neighbors, that is) it dies.

Definition 2.1. Starting from a given initial configuration, Life *terminates* if the sequence of successor configurations eventually stabilizes. This resulting

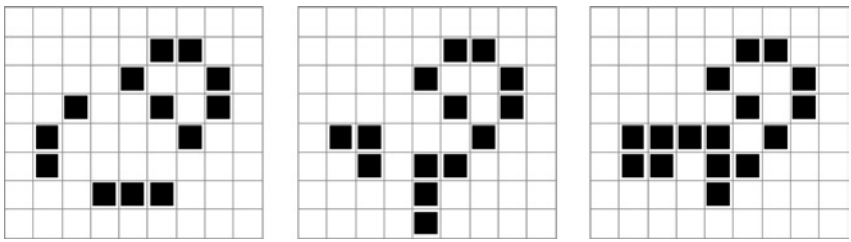

Fig. 1. Sample initial and two successor configurations of life.

configuration is called *rejecting* if it is empty (every cell `dead`), otherwise *accepting*.

An initial configuration is *finite* if, out of the infinite number of cells, only finitely many are `alive`.

Theorem 2.2. *Life with finite initial configuration is a system with infinite parallelism yet equivalent to the TM. More precisely:*

(i) *Given a finite initial configuration, its evolution through Life can be simulated by a TM.*

(ii) *There is a finite initial configuration capable of simulating the Universal (and thus any) TM.*

Simulation *here means: The TM terminates/accepts/rejects iff Life terminates/accepts/rejects.*

Proof: Life is easily programmed on a Turing Machine, for example by beginning with the source code for one of the many correct implementations of Life available on the internet. The converse Claim (ii) is a famous result based on a complicated construction; see, Berlekamp *et al*. (2004, Chap. 25) for details. □

In particular, Life matches but does *not* exceed the computing capabilities of a TM; cf. Atallah (1999, Section 26.4(1)). Here, finiteness of the initial configuration enters crucially of course. One can indeed show that *in*finite initial configurations in (ii) correspond to non-blank memory contents and thus to dropping in Observation 1.3 both Conditions (e) *and* (b).

2.2. Infinite Turing Concurrency

In order to focus on the power obtained from infinite parallelism *only* (that is, by removing just Condition (e), now consider the finite automata replaced by TMs. Indeed the three citations in Section 1.3. indicate that, whereas Quantum Computing—and in particular simulating a single classical TM (Benioff, 1980)—requires only a finite (or at most countably infinite) number of dimensions from quantum mechanical Hilbert Space, its *in*finitely many dimensions provide room for an infinite number of TMs: cf. Hilbert's Hotel.

Strictness of Chomsky's Hierarchy implies that a single TM is provably more powerful than a single automaton (Atallah, 1999, Section 25.3). One may therefore expect that the capabilities of an infinite number of TMs exceed those of an infinite number of automata (and thus actually lead to hypercomputation); by how much, however, turns out to depend.

In analogy to Definition 1.1., consider first the following notion of solving a problem by means of infinite parallelism.

Definition 2.3. Fix a problem $L \subseteq \mathbb{N}$ and a countably infinite family $(M_k)_{k\in\mathbb{N}}$ of TMs. This family *solves* L if

(i) each M_k, upon input of any $x \in \mathbb{N}$, eventually terminates and
(ii) for each $x \in \mathbb{N}$ it holds: $x \in L$ iff at least one M_k outputs "1" (accepts).

However observe that, whereas each individual M_k halts, the time required to do so may depend on k so that it takes infinitely long for the entire family $(M_k)_k$ to terminate. (We point out that this behavior resembles the *fair infinite nondeterminism* of van Emde *et al.* (1989).) In order to get the entirety of all their answers within finite time, the following additional requirement is therefore important:

(iii) Upon input of any $x \in \mathbb{N}$, all M_k terminate within finite time bounded independent[3] of k.

While seeming sensible at first glance, a second thought reveals that, even with this restriction, the resulting notion of 'infinitely parallel computability' is still unreasonable: simply because *any* problem $L \subseteq \mathbb{N}$ becomes trivially solvable by an appropriate family $(M_k)_k$. To this end let the program executed by M_k store the constant "1" if $k \in L$ and the constant "0" otherwise. Let its main part then operate as follows: Upon input of $x \in \mathbb{N}$ test whether $x = k$; if so, output the stored constant, otherwise output "0"; then terminate.

Observe that, in accordance with (iii), the test "$x = k$" can indeed be performed in time depending only on x but not on k by comparing only the inital segments up to the length of x. The trick lies of course in the according family $(M_k)_k$ solving L being shown to merely *exist*. More precisely, (i)–(iii) fail to require that the descriptions of all M_k and their constants be *computable*. Therefore, we finally include this condition as well, in Theoretical Computer Science known as uniformity.

(iv) A TM M_0 is capable of generating, upon input of k, (the encoding of) M_k.

Here, *encoding* refers to a 'blueprint' of M_k; formally: to its Gödel Number (Hopcroft *et al.*, 2001, Section 9.1.2).

2.3. Computational Power of Infinite Turing Concurrency

This section reveals that the Definition 2.3. (i)–(iv) indeed yields an interesting non-trivial way of hypercomputation. More precisely we show that, in this

[3] But of course depending on x—otherwise there would not be a chance for M_k to even *read* the input.

sense, infinite Turing-Parallelism can

- solve the Halting Problem H (Theorem 2.4)
- as well as Hilbert's Tenth Problem (Theorem 2.6)
- but *not* Totality. (Corollary 2.7)

While H refers to the question whether given TM M, started on a single given input x, eventually terminates, Totality asks whether M halts on *all* possible inputs. So in contrast to the first, this even more important property of correct software still remains intractable to automated verification even on this kind of hypercomputer.

Even more than already in the previous ones, proofs in this section regularly exploit well-known results from Theoretical Computer Science, for conciseness reasons just by indicating according references.

Theorem 2.4. *The Halting Problem is solvable by an infinity of TMs working in parallel in the sense of Definition 2.3. (i)–(iv).*

Proof: For each $k \in \mathbb{N}$, consider the TM M'_k working as follows: Given M and x, it simulates M operating on x the first k steps of if M halts within these steps, then M'_k outputs "1" and terminates; otherwise it outputs "0" and terminates. In other words, M'_k is basically a Universal Turing Machine $U(\cdot\,;k)$ with an additional counter for the number of steps simulated so far in order to abort as soon as this counter exceeds the prescribed threshold k.

Observe that the thus defined family $(M'_k)_k$ satisfies (i) and (ii) from Definition 2.3. Moreover, since M'_k is basically U with last argument fixed to k, one easily confirms that (iv) an appropriate TM M'_0 can indeed generate from k an encoding of this M'_k. And finally it is known (Du and Ko, (2000), Proposition 1.17, p. 25) that the simulation by a Universal TM is possible with at most quadratical overhead, i.e., M'_k can be achieved to have running time $t(n) \leq c \cdot (n \cdot k)^2$ for some constant c; Here, n denotes the length of the input to M'_k, that is, of the joint binary encodings of x and M.

Now let M_k be the TM obtained from applying the below *Linear Speed-Up Lemma* 2.5 to M'_k with $C := k^3$. It follows that M_k has running time bounded independent of k, that is, it complies with (iii) while still satisfying (i), (ii), and (iv).

In order to achieve Property (iii) in the above proof, the crucial ingredient is the below classical construction. In analogy to Moore's empirical law of technological progress, it basically says that a TM can be accelerated by any *constant* factor. □

Lemma 2.5. (Linear Speed-Up) *For each $C \in \mathbb{N}$ and any TM M' of time complexity $t(n)$, there exists another TM M simulating M' within running time $n + t(n)/C$.*

M can be obtained computationally from M'; i.e., there is a fixed further TM which, given an encoding of any M' and C, outputs an encoding of M as above.

Proof: See for instance, Atallah (1999, Theorem 24.5(b)). □

Recall that '*semi*-decidability' (also called *recursive enumerability*) weakens 'decidability' from Definition 1.1. in that, here, the TM is allowed in case $x \notin L$ to not halt but to loop endlessly (Hopcroft *et al.*, 2001, Section 8.2.5). By the first (and easy) part of Theorem 1.2, Hilbert's Tenth Problem is semi-decidable. More generally, every semi-decidable problem is the termination problem of an appropriate TM (Soare, 1987, Theorem II.1.2). Theorem 2.4 thus implies that infinite Turing-Parallelism can solve *any* semi-decidable $L \subseteq \mathbb{N}$. In fact, the converse holds as well:

Theorem 2.6. *A problem $L \subseteq \mathbb{N}$ is solvable in the sense of Definition 2.3. (i)–(iv) iff semi-decidable.*

Proof: By the above remark, it remains to consider the case that L is solvable by some parallel family $(M_k)_k$ according to Definition 2.3. We are going to semi-decide L on a single TM by means of the following simulation: Upon input of $x \in \mathbb{N}$ and for each $k \in \mathbb{N}$,

- obtain from M_0 a description of M_k by virtue of (iv)
- and simulate M_k on input x. (Observe its termination according to Property (i))
- If output is "1", halt; otherwise proceed with next k.

This algorithm indeed terminates iff at least one M_k outputs "1", that is (ii), iff $x \in L$. □

Corollary 2.7. *Even infinite Turing concurrency cannot solve Totality.*

Proof: Totality is well-known to *not* be semi-decidable. More specifically, we refer to (Soare, 1987, Theorem IV.3.2) where this problem is shown to be Π_2-complete, that is, [32, Definition IV.2.1 and Corollary IV.2.2] (Soare, 1987) reducible to $\overline{\emptyset^{(2)}} \notin \Sigma_2$, and therefore does not belong to the class $\Sigma_1 \subseteq \Sigma_2$ of recursively enumerable problems. □

3. CONCLUSION

Section 1 has pointed out that recent and independent approaches due to Kieu, Calude, and Pavlov to hypercomputation via quantum mechanics rely on some sort of infinite parallelism. Regarding the respective complicated intertwined quantum mechanical constructions, procedures, and analyses, we suggest bringing more clarity into this subject by considering algorithmic/computational issues separately from physical ones. This leads to the following two questions to be treated individually:

(1) Does quantum mechanics allow for infinite parallelism; and, if so, of what kind?
(2) What kinds of idealized infinite parallelism yield which computational power; that is, does it and by how far exceed the fundamental capabilities of a TM?

Section 2 contains answers to the second question. It reveals that in fact infinite *classical* (i.e., Turing-) parallelism is sufficient for solving both the Halting Problem as well as Hilbert's Tenth Problem. This leaves open whether the infinite dimensions of quantum mechanical Hilbert Space do indeed allow for this kind of infinite classical parallelism. Specifically,

- preparation of a certain initial state,
- its maintenance (in particular coherence) throughout the computational evolution, and
- read-out of the final result

are likely to raise here even more difficulties than already in the finite-dimensional case of 'standard' Quantum Computing (Gruska, 1999, Section 7.2). However (im-)practicality of hypercomputation should not be confused with (un-)existence, particularly in the light of Remark 1.4.

REFERENCES

Adamyan, V. A., Calude, C. S., and Pavlov, B. S. (2004). Transcending the limits of Turing computability. T. Hida, K. Saito, S. Si, (ed), Quantum Information Complexity. *Proceedings of Meijo Winter School 2003*, World Scientific, Singapore,pp. 119–137.

Atallah, M. J. (ed.) (1999). *Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton, Florida.

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2004). *Winning Ways for Your Mathematical Plays*, vol. 4, 2nd Edn., Academic Press, New York.

Beggs, E. J. and Tucker, J. V. (2004). Computations via Experiments with Kinematic Systems, Technical Report 5–2004, Department of Computer Science, University of Wales Swansea.

Benioff, P. (1980). The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing Machines. *Journal of Statistical Physics* **22**, 563– 591.

Bennett, C. H. and Landauer, R. (1985). The fundamental physical limits of computation. *Scientific American* **253**(1), 48–56.

Blass, A. (1984). Existence of bases implies the axiom of choice. *Axiomatic Set Theory, Contemporary Mathematics* **31**, 31–33.

Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers: $\mathcal{NP}$-completeness, recursive functions, and universal machines. *Bulletin of the American Mathematical Society* **21**, 1–46.

Burgin, M. and Klinger, A. (eds.) (2004). Super-recursive algorithms and hypercomputation. vol. 317. In *Theoretical Computer Science*, Elsevier, Amsterdam.

Calude, C. S., Dinneen, M. J., and Svozil, K. (2001). Reflections on quantum computing. In *Complexity*, Vol. 6(1), Wiley, New York.

Calude, C. S., Dinneen, M. J., and Peper, F. (eds.) (2002). Unconventional Models of Computation, Vol. 2509. In *Lecture Notes in Computer Science*, Springer, Heidelberg.

Calude, C. S. and Pavlov, B. (2002). Coins, quantum measurements, and Turing's barrier. In *Quantum Information Processing*, Vol. 1, Plenum, New York, pp. 107–127.

Copeland, J. (1997). The broad conception of computation. *American Behavioural Scientist* **40**, 690–716.

Copeland, J. (2002). Hypercomputation. *Minds and Machines* **12**, 461–502.

Copeland, J. and Proudfoot, D. (1999). Alan Turing's forgotten ideas in computer science. *Scientific American* **280**(4), 98–103.

Du, D.-Z. and Ker-I Ko (2000). *Theory of Computational Complexity*, Wiley, New York.

Eberbach, E. and Wegner, P. (2003). Beyond Turing Machines. *Bulletin of the European Association for Theoretical Computer Science* **81**, 279–304.

van Emde Boas, P., Spaan, E., and Torenvliet, L. (1989). Nondeterminism, fairness and a fundamental analogy. *The Bulletin of the European Association for Theoretical Computer Science* **37**, 186–193.

Etesi, G. and Németi, I. (2002). Non-Turing computations via Malament–Hogarth Space–Times. *International Journal of Theoretical Physics* **41**(2), 341–370.

Geroch, R. and Hartle, J. B. (1986). Computability and physical theories. *Foundations of Physics* **16**(6), 533–550.

Gödel, K. (1940). *The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis With the Axioms of Set Theory*, Princeton University Press, Princeton.

Gruska, J. (1999). Quantum Computing, McGraw-Hill, New York.

Hamkins, J. D. and Lewis, A. (2000). Infinite time Turing machines. *Journal of Symbolic Logic* **65**(2), 567–604.

Hogarth, M. L. (1992). Does general relativity allow an observer to view an eternity in a finite time?. *Foundations of Physics Letters* **5**(2), 173–181.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA.

Kieu, T. (2003). Computing the non-computable. *Contemporary Physics* **44**(1), 51–71.

Kieu, T. (2003). Quantum algorithm for Hilbert's Tenth Problem. *International Journal of Theoretical Physics* **42**, 1461–1478.

Matiyasevich, Y. V. (1970). Enumerable sets are diophantine. *Soviet Mathematics. Doklady* **11**, 354–357.

Ord, T. (2002). *Hypercomputation: Computing more than the Turing machine*, Honours Thesis, University of Melbourne, Melbourne; `http://arXiv.org/math.LO/0209332.`

Odifreddi, P. (1989). *Classical Recursion Theory*, North-Holland, Amsterdam.

Shagrir, O. and Pitowsky, I. (2003). Physical hypercomputation and the Church–Turing Hypothesis. *Minds and Machines* **13**, 87–101.

Soare, R. I. (1987). *Recursively Enumerable Sets and Degrees*, Springer, Heidelberg.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* **42**(2), 230–265.

Turing, A. M. (1939). Systems of logic based on ordinals. *Proceedings of the London Mathematical Society* **45**, 161–228.

Yao, A. C.-C. (2003). Classical physics and the Church–Turing Thesis. *Journal of the Association for Computing Machinery* **50**(1), 100–105.