Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 270 (1) (2011) 17–28

www.elsevier.com/locate/entcs

Computational Complexity in Non-Turing Models of Computation

The *What*, the *Why* and the *How*

Ed Blakey[1,2]

Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom

Abstract

We preliminarily recap what is meant by *complexity* and *non-Turing computation*, by way of explanation of our title, 'Computational Complexity in Non-Turing Models of Computation'.
Based on investigation of a motivating example, we argue that traditional complexity theory does not adequately capture the true complexity of certain non-Turing computers, and, hence, that an extension of the theory is needed in order to accommodate such machines.
We propose a framework of complexity that is not computation-model-dependent—that, rather, is extensible so as to accommodate diverse computational models—, and that allows meaningful comparison of computers' respective complexities, whether or not the comparison be with respect to different *resources*, and whether or not the computers be instances of different *models of computation*.
Whilst, we suggest, complexity theory is—without some modification—of limited applicability to certain non-standard models, we hope that the ideas described here go some way to showing how such modification can be made, and that members of the non-Turing-computation community—not least participants of *Quantum Physics and Logic/Development of Computational Models 2008*—find these ideas both useful and interesting.

Keywords: Computational complexity, non-standard/non-Turing computational models, precision.

This work forms part of the author's ongoing studies for his doctoral degree. A more complete account of this project can be found in [3], which is available at `http://users.ox.ac.uk/~quee1871/transfer.pdf`.

1 What. . .

. . . *do we mean by* complexity *and* non-Turing computation*?*

[1] We thank Bob Coecke and Joël Ouaknine (at Oxford) for their support, supervision and suggestions; Samson Abramsky and Peter Jeavons (at Oxford) for their useful comments about this project; Viv Kendon and colleagues (at Leeds) for useful discussions regarding the application of this work to quantum computing; Jonathan Mills (at Indiana, USA), Susan Stepney (at York) and others for their comments and suggestions made at Unconventional Computing 2007; and participants of the Second International Workshop on Natural Computing for their encouraging feedback and discussion.

[2] Email: `edward.blakey@queens.ox.ac.uk`

doi:10.1016/j.entcs.2011.01.003

In this section, we recall briefly what is meant by *computational complexity*, and note the concept's (understandable) bias towards the Turing machine. We also note, however, the widespread discussion—and, indeed, practical use—of other models, and ask whether more is needed of complexity theory in order to allow suitable analysis of non-standard computers.

1.1 Complexity

The field of *computational complexity* strives to categorize problems according to the cost of their solution. This cost is the *resource* (run-time, memory space or similar) consumed during computation; complexity theorists are interested particularly in the *increase* in resource as the computation's input grows.

So that the notions of complexity theory are well defined, the field has largely been developed *relative to a specific computation model*: the Turing machine. This choice is rarely seen as problematic: a vast majority of practical computation conforms to this model (specifically, real-world computations are typically performed by digital computers running programs that implement algorithms); the almost exclusive consideration of Turing machines is further bolstered—at least for those who overlook the distinction between computability and complexity—by the Church-Turing thesis [3]. Consequently, resource is virtually always taken to be a property—usually run-time—of a Turing machine (or algorithm, random access machine or similar). [4]

Because of the prevalence of digital computers, then, and because of the conjectured equivalence of the Turing machine to other computational models, this restriction of complexity theory to the exclusively algorithmic is seldom explicitly considered, let alone viewed as cause for concern. Indeed, the field is very successful, not only theoretically, but also practically, offering excellent guidance, for example, on the allocation of computational effort.

1.2 Non-Standard Computational Models

However, there has long been—and is today—an active community working on non-Turing forms of computer [5]:

- mechanical means by which differential/integral equations are solved (e.g., the Differential Analyzer; see [11]);
- the formation of soap bubbles between parallel plates, as used to find minimal-length spanning networks (possibly with additional vertices) connecting given vertices (see [17]);
- DNA-computing techniques that can tackle the directed Hamiltonian path problem (see [2]), amongst other graph-theoretic/combinatorial problems;

[3] The Church-Turing thesis is introduced in [12] and discussed in [8] and [20], amongst many others.

[4] Having said this, we acknowledge that consideration has been made—though in a largely ad hoc, and, hence, far from unified, way—of complexity in non-standard computational models. We note, however, that the true complexity of such computation is not always captured by such consideration; notably, this is true in the (circuit-model) quantum-computer case—see Sect. 3.1.

[5] We note also the growing recognition (see, e.g., [1] and [14]) of such computers' importance.

- standard (circuit-model [18]) and non-standard (adiabatic [15,16], measurement-based [19], continuous-variable [9], etc.) quantum computers;
- optical methods offering novel approaches to the Travelling Salesman problem (see [13]), etc.;
- the work of participants of *Quantum Physics and Logic/Development of Computational Models 2008*;
- and so on.

Despite the dominance of digital computers, non-Turing systems are of increasing importance.

Just as we can analyse the complexity of *algorithms* (and Turing machines, etc.), so we wish to be able to analyse the complexity of *non-standard computers*, not least so that we can compare the efficiency of our non-standard solutions with that of their digital counterparts. One may hope, then, that the complexity of non-Turing computers can be adequately analysed using the existing tools of traditional complexity theory.

We shall see that this is not the case: the Turing-machine-focused measures fail to capture the true complexity of certain systems.

2 Why. . .

. . . do non-Turing computers warrant different approaches to complexity analysis?

We claim above that traditional, Turing-based complexity theory overlooks the true complexity of some computing systems. In this section, we justify this claim by exhibiting just such a system and discussing its complexity.

2.1 Analogue Factorization System

We now outline an analogue system that factorizes natural numbers. The description given here is brief; for full details, see [4]. (The interested reader should also see [7], the pending US patent, of which the system is the subject, applied for by IBM and with the author as sole inventor, and [6], which describes a modified and in some senses improved version of the system. Finally, the system is used, as here, as a motivating example in [3].)

Just as with traditional algorithms, there is a practical limit to the size of numbers that the system can factorize; in contrast with traditional algorithms, however, the system suffers no increase in calculation time as the input number approaches this limit. Crucially for present purposes, the limit is not imposed by considerations (of run-time, memory space, etc.) made in traditional complexity theory, but rather by the increasing *precision* demanded of the user of the system. [6]

[6] We see here the first hints of a need for an extension of complexity theory.

Geometric formulation.

Note first that the task of factorizing natural number n has a geometric formulation: the task is exactly that of finding points both in the integer grid $\mathbb{Z}^2$ and on the curve $y = \frac{n}{x}$ (the coordinates of such a point are two of the sought factors). Since factors of n are members of $\{0, \dots, n\}$ (in fact, of $\{1, \dots, n\}$), we need search for grid/curve points (x, y) only in the region $0 \leq x, y \leq n$; and since (x, y) reveals the same factors as (y, x), we need search only in the region $0 \leq x \leq y \leq n$. The curve $y = \frac{n}{x}$ is a conic section (specifically, a hyperbola), and so is the intersection of the (x, y)-*plane* and a *cone*; the factorization method's implementation exploits this fact.

Implementation of the grid.

We implement the part of the *integer grid* that is of interest (i.e., that lies in the region identified above) using a source S (of wavelength $\frac{2}{n}$) of transverse waves, reflected by mirrors M_1, M_2 and M_3 so as to produce a certain interference pattern; the points of maximal wave activity in this pattern model integer grid points, with maximally active point $\left(\frac{a}{n}, \frac{b}{n}, 0\right)$ modelling grid point (a, b) (where $a, b \in \mathbb{Z}$).[7] Since the wavelength of radiation from S depends on n, its being set forms part of the computation's input process.

See Fig. 1 for the apparatus used in implementing the integer grid (in which B is a black body that absorbs some radiation from S), Fig. 2 for the route of propagation of a sample ray[8], and Fig. 3 for the maxima (shown as dots) of the resultant interference pattern within the region $R := \{(x, y, 0) \mid 0 \leq x \leq y \leq 1\}$ that, since $\left(\frac{a}{n}, \frac{b}{n}, 0\right)$ models (a, b), models the region $0 \leq x \leq y \leq n$ of interest (this figure takes as its example $n = 5$).

Implementation of the cone.

The *cone* is implemented by a source P_n of radiation (the vertex of the cone) and a part-circular sensor C_n (the circle is a cross section of the cone).[9] See Fig. 4. The subscripts reflect the fact that the positions of P_n and C_n depend on n; the positioning of these components forms part of the input process for the computation.

Interpreting output.

Having described the apparatus and alluded to the input method (namely, setting to the appropriate, n-dependent values the wavelength of S and positions of P_n and C_n), we turn to the interpretation of output. Radiation arriving from P_n at a point on C_n displays high-amplitude interference (due to the pattern of waves

[7] In fact, these maximally active points model those integer-grid points *that have coordinates of the same parity*. This is sufficient provided that we assume n to be odd, for then any factors of n are odd. (Should a factorization be required of an even number, it is computationally trivial—in the Turing-machine realm—iteratively to divide by two until an odd number, which can be factorized as described here, is obtained.)

[8] Note that the ray is reflected back along itself by M_3; this produces a *standing wave*.

[9] It is proven in [4] that the curve of C_n is the circular arc produced by projecting from P_n the curve $G_n := \left\{(x, y, 0) \in R \mid \frac{1}{xy} = n\right\}$ onto the plane $y = 2 - x$. Hence, radiation arriving from P_n at a point on C_n passes through the plane $z = 0$ at a point $(x, y, 0)$ such that $\frac{1}{xy} = n$.

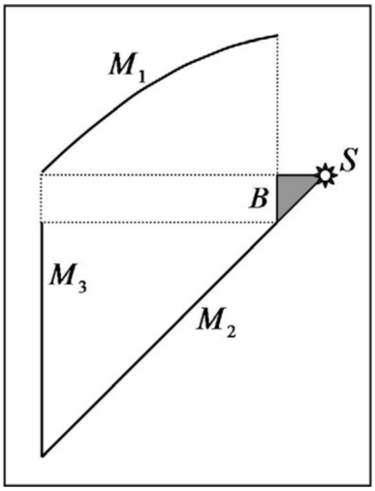

Fig. 1. The layout of the apparatus that implements the integer grid.

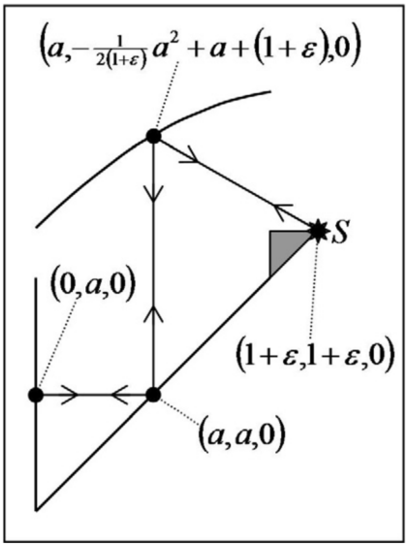
$\left(a, -\frac{1}{2(1+\varepsilon)}a^2 + a + (1+\varepsilon), 0\right)$
S
$(0, a, 0)$
$(1+\varepsilon, 1+\varepsilon, 0)$
$(a, a, 0)$

Fig. 2. The path of a ray propagating from S via the three mirrors M_i back to S.

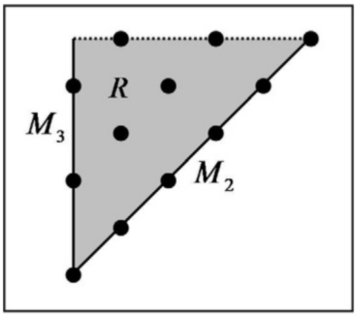

Fig. 3. The maximum amplitude points in the region R, where $n = 5$.

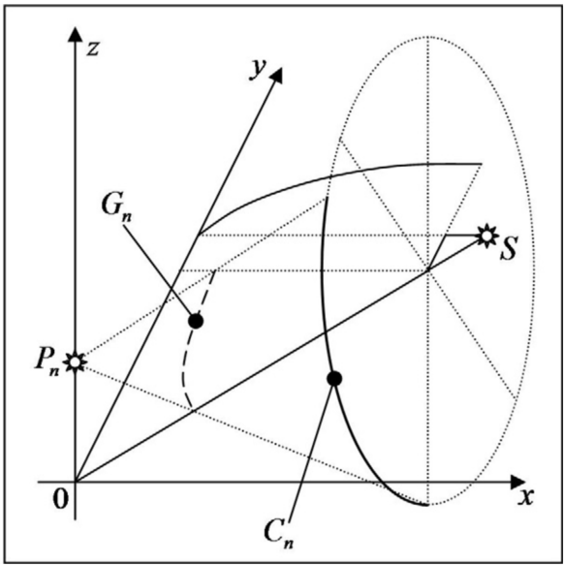

Fig. 4. The layout of the apparatus that implements the cone.

from S) if and only if the point $(x, y, 0)$ at which it meets the (x, y)-plane models an integer point; that is, if and only if $\frac{1}{x}, \frac{1}{y} \in \mathbb{Z}$. Further, by construction of P_n and C_n, $\frac{1}{x} \cdot \frac{1}{y} = n$ (see footnote 9). Hence, this radiation displays high-amplitude interference if and only if $\frac{1}{x}$ *and* $\frac{1}{y}$ *are factors of* n. Thus, to interpret the results, we measure the coordinates of a point (that which displays high-amplitude interference) on C_n and convert them into those of a point (that through which the ray passes) in the (x, y)-plane. Explicitly, radiation from P_n incident on a point $(a, 2-a, c)$ on C_n passes through $\left(\sqrt{\frac{a}{n(2-a)}}, \sqrt{\frac{2-a}{na}}, 0\right)$; hence, if the radiation arriving from P_n at $(a, 2-a, c)$ on C_n displays high-amplitude interference, then $\left(\sqrt{\frac{a}{n(2-a)}}, \sqrt{\frac{2-a}{na}}, 0\right)$ models an integer point on the hyperbola under consideration: $\sqrt{\frac{n(2-a)}{a}}$ *and* $\sqrt{\frac{na}{2-a}}$ *are factors of* n; conversely, all factors have such a point on C_n. Having set up the apparatus as described here, then, the factors of n are so found.

We have outlined an analogue factorization system. [10] It offers much-improved run-times when compared with existing, algorithmic solutions, largely because it directly, physically implements the problem rather than converting the problem into a contrived instance of the standard computation model. The *polynomial* time and space complexities [11] serve, however, to highlight not the power of the system but the incompleteness of traditional complexity theory. As n increases, the system *does* require exponentially more resource (though neither specifically time nor space); in particular, the *precision* with which n must be input (by setting the wavelength of S and the positions of P_n and C_n) and its factors read (by measuring the positions of

[10] We reiterate that [4] offers a more complete description of the system; it also offers proof of the system's correct functioning.

[11] These are discussed further in the following section, and formalized in [3].

points on C_n) increases exponentially with n. This suggests that, for some computers, traditional, 'algorithmic' complexity theory is inadequate; we aim to introduce notions of complexity more suitable in such contexts.

2.2 Complexity of the System

Using the system.

The use of the system to factorize n consists of:

(i) calculation of the values $\frac{2}{n}$ (to be used as the wavelength of S) and $\sqrt{\frac{2}{n}}$ (to be used as the z-coordinate of P_n and of the centre of the circle of C_n);

(ii) supply of n to the system, by adjusting the wavelength of S and the height (z-coordinate) of P_n and C_n in accordance with the values found during (i);

(iii) interference of the radiation in the system, which entails propagation of the radiation over a fixed distance (since the same apparatus is—bar the adjustments made in (ii)—used for all values of n);

(iv) measurement of the positions of high-amplitude interference points on the sensor C_n; and

(v) conversion of the positions measured during (iv) into factors: $(a, 2-a, c) \mapsto \left(\sqrt{\frac{n(2-a)}{a}}, \sqrt{\frac{na}{2-a}}\right)$.

We consider now the computational complexity, with respect to several resources (formally defined in [3]), of these steps of the factorization system's use.

Time complexity.

Consider first *time*; the time complexity of the system, we claim, is polynomial in the size (i.e., number of digits) of the input value.

Note that, in the Turing-machine realm, steps (i) and (v) above take only polynomially long (in the size of n), the former since $\frac{2}{n}$ and $\sqrt{\frac{2}{n}}$ need be calculated only with sufficient precision that n can be retrieved *given that* $n \in \mathbb{Z}$, and the latter since sought values $\sqrt{\frac{n(2-a)}{a}}$ and $\sqrt{\frac{na}{2-a}}$ are integers. Note further that steps (ii) – (iv) take a constant time: larger values of n take no longer to process in accordance with these stages. Notably, step (iii), during which the actual factorization is performed, takes constant time; compare this with known algorithmic methods, where computation time increases exponentially with the size of n (see, e.g., [10]). Thus, the time complexity of the system as a whole is, as claimed, *polynomial* in the size of the input.

Space complexity.

Similarly, when considering the resource of *space*, [12] we see that only the Turing-machine calculations of steps (i) and (v) (which essentially prepare input and interpret output) consume an increasing volume as n increases, and these only a *polynomially* increasing volume (in the size of the input); and for steps (ii) – (iv), the same, fixed-size apparatus—occupying the same, fixed space—is used for all values of n (though the positions of P_n and C_n depend on n, there exists a finite, n-independent, bounding cuboid in which the apparatus lies *for all* n). Thus, the space complexity of the system is *polynomial* in the size of the input.

The resources of time and space are arguably of paramount relevance when considering instances (Turing machines, random access machines, etc.) of the standard, algorithmic computational model. Notions of complexity developed with only these instances in mind, however, are understandably poor at capturing the complexity of instances of wildly different models; the factorization system above does indeed have polynomial time and space complexities, and yet *does* require exponentially increasing resource as n increases. Notably, larger values of n require *exponentially increasingly precise manipulation* of the input parameters (the wavelength of S and the positions of P_n and C_n) and *exponentially increasingly precise measurement* of the output parameters (the coordinates of points on C_n), and there is no reason for which we should not view *required precision* as a resource. Accordingly, we consider now the *precision complexity* of the system, which is certainly not polynomial, and hence better captures the system's true complexity than do the resources of time and space.

Precision complexity.

The intention of *precision complexity* is to capture the lack of robustness against input/output imprecision of a 'physical' (e.g., analogue/optical/chemical) computer. We consider the example of setting the wavelength in our factorization system; other input parameters (the positions of P_n and C_n) and output parameters (the positions of points on C_n), which we omit for brevity, [13] can be analysed similarly. (A formal account/definition of precision complexity is given in [3].)

Given n, which we wish to factorize, we intend to set the wavelength to $\frac{2}{n}$. In practice, due to technological limitations on our control over the wavelength (we have in mind imprecise use of a variable resistor or similar), we may set the wavelength to $\frac{2}{n'}$, which we know only to lie in $\left[\frac{2}{n} - \epsilon, \frac{2}{n} + \epsilon\right]$ for some real error term ϵ. However, we may engineer the system (using standard analogue techniques) so that non-integer input values are rounded off so as to correct 'small' errors: given wavelength $\frac{2}{x}$ (for arbitrary $x \in \mathbb{R}$), the value to be factorized is taken to be $\left\lfloor x + \frac{1}{2} \right\rfloor$. So, provided that the supplied wavelength $\frac{2}{n'} \in \left[\frac{2}{n} - \epsilon, \frac{2}{n} + \epsilon\right]$ falls in the interval

[12] Space, as traditionally encountered with Truing machines, etc., can be viewed as the storage capacity of the memory required by a computation; we consider the analogous notion of required *physical volume*.

[13] The omission, whilst partly for brevity, also reflects the redundancy of consideration of other parameters' precision requirements: once the setting of the wavelength has been shown to require exponentially increasing precision, then we have that the *overall* precision complexity is exponential, regardless of the contribution from other parameters.

$\left(\frac{2}{n+\frac{1}{2}}, \frac{2}{n-\frac{1}{2}}\right]$ of values 'corrected' to n—i.e., *provided that* $\epsilon < \frac{1}{n\left(n+\frac{1}{2}\right)}$—, then the system processes the correct input value. Note that this constraint on ϵ implies that the precision required of us when setting the wavelength increases *quadratically* with n, and hence *exponentially* with the size of n.

We see, then, that the system's *precision complexity*, which we informally introduce here and, we iterate, treat formally in [3], is exponential, and therefore of much greater significance than the system's polynomial time/space complexity. *The significant complexity measure is overlooked by traditional complexity theory.*

3 How...

...should we measure *non-Turing computers' complexity, and how can such complexity be meaningfully* compared *with that of Turing machines?*

There are two aspects to the way in which, we suggest, complexity in non-standard computation should be analysed.

- First, we advocate consideration of more diverse selections of *resources*; for example, we see above that not only traditional, algorithmic measures such as time and space, but also more physical measures such as precision, should be considered. See Sect. 3.1.
- Secondly, we should like to be able to *compare* in a meaningful way the respective complexities of instances of different computational models, with respect to different complexity resources; we accordingly advocate use of the notion of *dominance*, which offers a criterion that tells us which of a computation's resources are 'relevant'; once such are identified, they can be compared within the usual '$\in \mathcal{O}$' pre-ordering. See Sect. 3.2.

3.1 Resources in Non-Turing Computation

We note above the following:

- that traditional complexity theory and the resources (time, space, etc.) considered therein are inspired by the Turing-machine model of computation, almost totally to the exclusion of other models;
- that, while this is clearly adequate when considering Turing machines, certain non-standard computers (quantum computers being a timely example) are inadequately catered for by these resources; and
- that it is possible (and even natural, once the problem has been acknowledged) to define resources (e.g., precision) that better capture the true complexity of non-Turing computers.

The solution is obvious, then: when working with a non-standard computational model, we should consider which resources—both algorithmic and non-algorithmic—are consumed during computation, and should explicitly measure the complexity of our computation with respect to these resources; this gives a more complete picture, and more confidence in our understanding, of the computation's complexity.

The need for consideration of different resources is particularly evident in the *quantum-computer* case. An arbitrary algorithm (or Turing machine or similar) can, by definition of complete, be expressed as a conversion of input to output via operations exclusively taken from some complete set of what are deemed to be 'atomic' operations. For a given input value, the number of such operations performed during this conversion is an accurate measure (or, depending on viewpoint, a definition) of run-time. Similarly, an arbitrary *quantum* computation can be expressed as the preparation of several quantum bits, followed by a sequence of applications to subsets of these quantum bits of 'atomic' unitary operations taken from a complete set, followed by a measurement of the system. As in the classical case, an enumeration of the invocations of these atomic operations gives a measure of the system's complexity; indeed, this is the basis of an existing definition of complexity of circuit-model quantum computing devices (see [18]). Also as in the classical case, however, the result is essentially a measure of run-time, which is not, we suggest, particularly relevant to quantum systems.[14]

By introducing and considering new resources, specifically ones similar to precision, we may, we suggest, better encapsulate the true complexity of a quantum system; this complexity arises, after all, because of our limited ability to take precise measurements from the system.

3.2 Comparing Complexity

Whereas reasoning directly about the computational complexity of a *problem* seems inherently difficult, it is relatively easy (once appropriate resources are considered) to ascertain the complexity of specific *methods* (algorithms, analogue computers, Turing machines, etc.) *that solve the problem.* Consequently, our sole understanding of a problem's complexity is, in the majority of cases, gleaned via our having determined the complexity of solution methods for the problem; specifically, we have that the problem's complexity is bounded from above by the complexity of the most efficient known solution method.

In order to improve such bounds, it is desired to consider as large a set as is possible of solution methods for a problem. Each set so considered in practice, however, is likely to be of solution methods taken from a single model of computation (often that of the Turing machine). This is a necessary evil of our inability meaningfully to compare the complexity of instances of different computation models.

Required, then, is a more general framework in which to study complexity; in particular, we wish to be able to use the framework to consider on a consistent and comparable footing the complexity—with respect to several notions of resource—of instances of diverse models of computation; only when we can meaningfully

[14] The advantage of quantum computers over their classical counterparts arises primarily from the use of entangled states and the effective parallelism that such use allows; a disadvantage is the strictly constrained way in which information can be read from the quantum system. The run-time of such a system, then, is a reflection of neither the 'amount of computation' being performed (due to the parallelism) nor the 'difficulty' in using the system (which chiefly arises during measurement).

compare, say, the respective complexities of a Turing machine and a DNA computer can we begin to consider larger, model-heterogeneous sets of solution methods, and hence obtain improved bounds on the complexity of problems. Accordingly, we introduce the notion of *dominance* of resource.

3.2.1 Dominance.

Recall that, in the complexity analysis of the factorization system above, we suggest that precision, on which the system has an exponential dependency, is more relevant than either time or space, on each of which it has polynomial dependency. This can be formalized (and a general concept abstracted) by noting that $T, S \in \mathcal{O}(P)$, but neither $P \in \mathcal{O}(T)$ nor $P \in \mathcal{O}(S)$, where P, T and S stand respectively for the precision, time and space complexity functions; we say that, relative to $\{P, T, S\}$, precision is *dominant*. More generally, relative to a set $\{X_1, \dots, X_k\}$ of complexity functions (with respect to respective resources $x_1, \dots, x_k$), resource x_i is *dominant* if and only if $X_i \in \mathcal{O}(X_j) \Rightarrow X_j \in \mathcal{O}(X_i)$ for all j.

By considering for a given solution method (e.g., Turing machine/analogue computer/quantum system) only the *dominant* resources, we focus on the relevant measures for the method—dominance formalizes a resource's relevance: resources that are dominant impose the asymptotically greatest cost, to the extent that non-dominant resources may be disregarded as irrelevant. Further, we can compare solution methods according to their relevant (i.e., dominant) resources (using the '$\in \mathcal{O}$' pre-ordering). We therefore have a framework in which can be made meaningful and consistent comparisons of computation-model-heterogeneous sets of computers; the framework can accommodate instances of various models of computation, and provide structure according to cost in terms of various resources.

(This framework and the notion of dominance on which it is based are investigated in greater detail in [5].)

4 Conclusion

Complexity theory has developed with a bias towards the Turing-machine model. This is readily explained, and poses no problem in 'standard' use; we note, however, that the field is—without some modification—of limited applicability to non-standard models. We hope that the ideas described here (which are explored more fully in [3]) go some way to showing how such modification can be made, and that members of the *QPL/DCM 2008* community find these ideas both useful and interesting.

References

[1] Adamatzky, A. (editor), Int. J. of Unconventional Computing, Old City Publishing (2005 onwards)

[2] Adleman, L. M., *Molecular Computation of Solutions to Combinatorial Problems*, Science **266** (1994), 1021–1024

[3] Blakey, E., *A Model-Independent Theory of Computational Complexity; Price: From Patience to Precision (and Beyond)*, (2008)

[4] Blakey, E., *An Analogue Solution to the Problem of Factorization*, Oxford University Computing Science Research Report CS-RR-07-04 (2007)

[5] Blakey, E., *Dominance: Consistently Comparing Computational Complexity*, (in production)

[6] Blakey, E., *Factorizing RSA Keys, an Improved Analogue Solution*, Proceedings of the Second International Workshop on Natural Computing, Nagoya, Japan (to appear) (2007)

[7] Blakey, E., *System and Method for Finding Integer Solutions*, US patent application 20070165313 (2007)

[8] Bovet, D. P. and P. Crescenzi, "Introduction to the Theory of Complexity", Prentice Hall (1994)

[9] Braunstein, S. L. and A. K. Pati, "Quantum Information with Continuous Variables", Springer-Verlag (2003)

[10] Brent, R. P., *Recent Progress and Prospects for Integer Factorisation Algorithms*, LNCS **1858** (2000), 3–20

[11] Bush, V., *The Differential Analyzer: a New Machine for Solving Differential Equations*, J. of the Franklin Inst. **212** (1931), 447–488

[12] Church, A., *An Unsolvable Problem of Elementary Number Theory*, Am. J. of Math. **58** (1936), 345–363

[13] Haist, Tobias and Wolfgang Osten, *An Optical Solution for the Traveling Salesman Problem*, Optics Express **15** no. 16 (2007), 10473–10482

[14] Hoare, C. A. R. and R. Milner (editors), Grand Challenges in Computing, The British Computer Society (2004)

[15] Kieu, T. D., *Hypercomputation with Quantum Adiabatic Processes*, TCS **317** (2004), 93–104

[16] Kieu, T. D., *Quantum Adiabatic Algorithm for Hilbert's Tenth Problem*, arXiv:quant-ph/0310052 (2003)

[17] Miehle, W., *Link-Length Minimization in Networks*, Operations Research **6** no. 2 (1958), 232–243

[18] Nielsen, M. A. and L. Chuang, "Quantum Computation and Quantum Information", CUP (2000)

[19] Raussendorf, Robert, Daniel E. Browne and Hans J. Briegel, *Measurement-Based Quantum Computation on Cluster States*, Physical review A **68** no. 2 (2003)

[20] Sipser, M., "Introduction to the Theory of Computation", PWS (1997)