Nat Comput (2011) 10:1245–1259
DOI 10.1007/s11047-010-9226-9

Unconventional complexity measures for unconventional computers

Ed Blakey

Published online: 3 November 2010

Abstract Unconventional computers—which may, for example, exploit chemical/analogue/quantum phenomena in order to compute, rather than electronically implementing discrete logic gates—are widely studied in both theoretical and practical contexts. One particular motivation behind unconventional computation is the desire efficiently to solve classically difficult problems—we recall chemical-computer attempts at solving NP-complete problems such as the Travelling Salesperson Problem—, with computational complexity theory offering the criteria for judging this efficiency. However, care must be taken here; conventional (Turing-machine-style) complexity analysis is not always appropriate for unconventional computers: new, non-standard computational resources, with correspondingly new complexity measures, are often required. Accordingly, we discuss in the present paper various resources beyond merely time and space (and, indeed, discuss various interpretations of the term 'resource' itself), advocating such resources' consideration when analysing the complexity of unconventional computers. We hope that this acts as a useful starting point for practitioners of unconventional computing and computational complexity.

Keywords Complexity · Computational resources · Factorization · Precision · Unconventional computation

1 Background

1.1 Computational complexity theory

Computation—whether performed by a Turing machine (see Turing 1936), a real-life digital computer, a quantum system, or any other system that has provision for accepting input and providing corresponding output—must be *efficient* to be of practical use. The question of which parameters and attributes of a computation have a bearing on its

E. Blakey (✉)
Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
e-mail: edward.blakey@queens.ox.ac.uk

efficiency or otherwise, and of which values of these parameters and attributes confer this efficiency, is to some extent context-dependent (for example, portable but non-critical devices may favour space- over time-efficiency, whilst supercomputers processing meteorological data may not), though a widely accepted and highly formalized criterion for efficiency is offered by *computational complexity theory*.

Complexity theory suggests, and decades of practical experience corroborate, that a computer's efficiency corresponds to its using only a *polynomial* amount of computational resources (this polynomial function is of the *size* of the computation's input value). Since complexity theory has been developed primarily with the Turing machine and equivalent models/paradigms of computation in mind, these resources (that we wish to scale polynomially so as to enjoy efficiency) are traditionally *run-time* and *memory space*; this limited view of resource is adequate when analysing the complexity of *standard* computers such as the (non-parallel, deterministic) Turing machine and closely related physical instantiations such as the digital computer, but (as we see in Sect. 1.3) can lead to an unrealistically optimistic measurement of the complexity of *non-standard*, *unconventional* computers. The problem is that unconventional computers (for more about which see Sect. 1.2) may well consume unconventional resources (that is, those other than time and space), and the complexity measures corresponding to these resources should therefore be, though often are not, considered.

1.2 Unconventional computation

As we hint at above, traditional complexity theory is adequate for analysis of Turing-machine-style computers, but does not necessarily cater sufficiently for unconventional computers. Whilst it is unnecessary in the context of the present journal to describe or define unconventional computers (in their many forms: quantum, chemical/DNA, analogue, kinematic, optical, slime-mould, etc.), it is important to mention that *such computers consume resources other than time and space*, and that *these resources may contribute significantly to the system's complexity* in that they may impinge on the system's efficiency before availability of time or space has become pressing. It is at least prima facie feasible, for instance, that a kinematic computer will need *energy* to compute, and that the time and space complexities of the system may be dwarfed by its 'energy complexity'; or that an analogue computer requires of the user a certain *precision* in order to function correctly, and that 'precision complexity' is more indicative of the system's efficiency than its time and space complexities. We consider now the latter example in more detail.

1.3 Motivating example

We recall from Blakey (2008c) an analogue/optical system for factorizing natural numbers. Whilst we do not describe the system in detail here (a full account is available in Blakey (2008c), and an earlier, related system is described in Blakey (2007a) and covered by patent (Blakey 2008d)), we note now some important features: the system can factorize numbers using time and space *polynomial* in the numbers' size (this is in contrast with the *exponentially* scaling time required by known algorithmic solutions to the problem of factorization—see Brent (2000)[1]), but the precision with which the user must manipulate

[1] Although (Brent 2000) is some years old now, the stated exponential time requirement remains state-of-the-art.

and measure certain physical parameters (so as to effect input to and output from the system) increases as an *exponential* function of the size of the number to be factorized.

For present purposes, then, the relevant point is that *the true complexity of the system and the bars to its practical efficiency arise because of issues relating to the resource of* **required precision** *rather than of* **time** *or* **space**. This exemplifies the fact that insightful analysis of the complexity of unconventional computers entails consideration of unconventional resources. We now consider several such resources.

2 Commodity resources

2.1 What we mean by 'resource'

Note that, for present purposes, *resource* has a more specific meaning than its common usage[2] suggests. 'Resource' could validly be taken to mean not only those *commodities* consumed/required during computation (i.e. "when necessary" as in the quotation in Footnote 2)—time, space, precision, etc.—, but also the presence of non-determinism, the use of oracles, etc. (that is, features of the computational model/enveloping laws of physics, which features apparently augment computational power), the costs involved in manufacturing (rather than running) a computer (including, for example, manufacture of any entangled/cluster states from which respectively a general/measurement-based quantum computation proceeds; and including manufacturing costs viewed as thermodynamic costs—see Zurek 1989; Lloyd and Pagels 1988), etc. We may, further, consider 'information-theoretic' resources and the inequalities (based essentially on simulation power) therebetween; similar ideas arise in the context of trade-offs between (computation-time, 'commodity') resources.

In the present paper, we are concerned primarily with what we call *commodity resources*. These are computational resources consumed or required by a computer during execution of a computation (as opposed, for example, to resources used during a computer's manufacture, and to 'information-theoretic' resources such as communication channels; see Sect. 3 for more on non-commodity resources). Commodity resources include, but are by no means limited to, time and space.

For the remainder of Sect. 2, by 'resource' we mean 'commodity resource'.

An important question here is:

(★) *what methods can be used to identify the relevant resources for a given computational model?*

This question may be posed either in generality—'given an *arbitrary* computational model, which resources should be considered?'—or for specific models; we consider both levels of generality (see Sects. 2.3 and 2.6 respectively).

Some clarification is needed on what it means for a resource to be 'relevant'. An intuitive understanding of this concept can be gleaned from the analogue factorization example of Sect. 1.3, which, we have commented, has time and space complexities (both polynomial) that fail to capture the system's actual, *exponential* complexity; the unconventional resource of *precision* captures the true complexity here, and, hence, is *relevant* (unlike time and space).

[2] We find as a dictionary definition of 'resource' the following: "[a] means of supplying some want or deficiency; a stock or reserve upon which one can draw when necessary" (OED 1989).

Note that, whereas the phrasing of our question (★) suggests that resources' relevance is a function of the *computational model* under consideration, it may in fact vary between *instances* of the model: one computer of a certain model may, say, have time but not precision as a relevant resource, whilst, to another of the same model, precision but not time may be relevant. Nonetheless, when identifying *candidate* relevant resources, the choice of model seems more influential than the choice of specific computer. The complexity functions for these candidates (corresponding to the model(s) under consideration) can then be evaluated for the specific computer(s) of interest to see which are relevant (using notions such as *dominance*, which is defined and described in Blakey (2008b)).

Our focus is, of course, on *unconventional* models of computation. We claim as an aside that, in the conventional, Turing-machine model, the resources of *time* and *space* (and variations thereon) encompass all relevant complexity behaviour. We recall, for example, Păun's belief (see Păun 2008) that "[t]he standard dimensions of computations are time and space"[3]; neither do we need to take Păun's word for it: we note that many standard complexity classes are defined in terms of what is achievable by various computers within a certain *time* (loosely, these are P, NP (including the subclass of NP-complete problems), coNP, PH, EXP, NC, P/poly, BPP, BQP, PP) or using a certain amount of *space* (loosely, PSPACE, AC^0, NC, L).[4]

(Of course, should one deem parallel computation to be conventional, then to the conventional resources of *time* and *space* must we add *number of processors*. However, one may—and we do—view parallelism as an issue separate from our notions of (unconventional) complexity theory: susceptibility to a parallel approach is a property of a *problem*; of more interest here is how efficiently each parallel sub-computation can be performed (whether by Turing machine or unconventional computer); then account of parallelism is taken simply by summing respective sub-complexities. Similarly to parallelism, probabilism may be—but here is not—viewed as a conventional computational practice, whence the resource *number of random bits* (cf. Sect. 3.2) must be mentioned in a list of conventional resources).

2.2 Formalizing commodity resources

We model *commodity resources* (denoted by A, B, etc.) as functions that depend upon the choice of *computer* and that map each *input value* to the corresponding amount[5] of *resource* used by the computer in processing the input value.

So, where Φ is a computer and x an input value for Φ, $A_\Phi(x)$ (or simply $A(x)$ if the choice of Φ is understood) is the amount—in fact, *number of units*: we stipulate that resource A have codomain $\mathbb{N} \cup \{\infty\}$ (the natural numbers[6] augmented with an infinity element)—of resource A consumed by Φ in processing x.

[3] Păun adds that "[t]his is true for computer science, not necessarily for the brain"; not wishing to deny the brain recognition as a computer, we must, and here do, question what resources are involved in unconventional computation.

[4] The 14 classes mentioned here are those in the 'Petting Zoo' section of Scott Aaronson's Complexity Zoo (Aaronson 2005) (ignoring classes of function problems)—purportedly the most important (i.e., most referenced/fundamental/etc.) classes; those in the Petting Zoo but not amongst the 14 are MA, AM, SZK, which nonetheless are not defined in terms of resources other than time and space.

[5] This amount may, when no finite amount of a resource is sufficient for a computation to complete satisfactorily, be infinite (e.g., no finite amount of the resource of *time* suffices when a Turing machine has entered an infinite loop).

[6] Note that, in the present context, zero is a natural number: $\mathbb{N} := \{0, 1, 2, \ldots\}$.

With each resource is associated a *complexity function*. For a given computer and resource, we may consider how the resource scales: we may be interested not in the resource required by the computer in processing *some specific input value* (i.e., in some '$A(x)$'), but in the resource required *as a function of the input value's size*[7]—this is the *complexity function* corresponding to the resource. Specifically, we make the following definition.

Definition 1 (complexity function) *Let* Φ *be a computer with set* X_Φ *of possible input values and let A be a resource. The complexity function* A^*_Φ*, corresponding to resource A, is given by*

$$A^*_\Phi(n) := \sup\{A_\Phi(x) \mid x \in X_\Phi \wedge |x| = n\}.$$

Whilst A, B, etc. denote types of *resource*, then, A^*, B^*, etc. denote types of *complexity*.

(Note that 'resource' has not been *defined*; we have described necessary, but not sufficient, properties of the notion, and consider further restrictions in Sect. 2.4, where the notion is summarized.)

2.3 Model-independent resources

We look first at some resources that apply to all computational models.

2.3.1 Time and space

Run-time and *memory space* are the standard resources considered in complexity analyses of Turing machines; indeed, up to variations (such as ink, head reversals, etc.), they are the only ones—recall the discussion of Sect. 2.1.

Whilst we see above that consideration of unconventional computation necessitates consideration of unconventional resources, this is not to say that time and space are not still relevant in unconventional contexts. Furthermore, it is clear for virtually all physical computing paradigms how to generalize these two resources from Turing machines to the wider class of physical computers. *Time* can be taken to be the number of units of physical time (measured in seconds, say) elapsed during a computation (i.e., between input and output) performed by a physical system[8]; *space* can be taken to be the physical volume (in cubic metres, say) occupied by the system (including any required storage space, electrical or not).

2.3.2 Material cost

In addition to the material cost of *constructing* a computer (which may, from a commodity-resource point of view, be supposed to be a constant, one-off cost; or which can be treated

[7] We defer to Blakey (2008a), Blum et al. (1997) and standard complexity theory further discussion of *size functions* (that map input values to their non-negative, real sizes), noting here by way of illustration only that the size of a natural number n expressed in place notation—with base b, say—can be taken to be the number of digits of n (or the often-convenient approximation $\log_b(n)$). We denote the size of an input value x by $|x|$.

[8] Here we implicitly assume that computations are not affected by relativistic effects, which is clearly not the case with relativistic computation; for this model, then, our definition of time needs modification—see Sect. 2.6.

as a non-commodity resource—see Sect. 3.1), there may be a cost in *running* it (over and above energy costs, which we discuss in Sect. 2.6 below). For example, if memory is implemented in such a way that each write (either to a fresh or used memory cell) costs a constant amount, then we may wish to consider the resource of 'ink'; this existing notion can be generalized to the resource of *material cost*.

2.3.3 Thermodynamic cost

Strictly speaking, this resource is not applicable to computers from *all* paradigms (failing, in particular, for those from abstract, mathematical models), but is at least applicable to those that are *physically implemented*; Turing machines, then, are excluded, but digital computers that implement Turing machines are not.

The idea behind this resource is that computation typically *erases* information: evaluating a function (which will not in general be injective) in such a way that the input is destroyed and only the output is available after computation represents a loss of information, an increase in entropy and, hence, a thermodynamic cost; this concept was introduced by Landauer (1961) and developed by Bennett and Vitányi (who survey the notion in Bennett (1982) and Vitányi (2005) respectively) amongst others. Zurek (1989)—in which the corresponding complexity measure is explicitly introduced—notes that limits on the thermodynamic cost (a form of *computational* complexity) of a computation can be inferred by considering its *algorithmic* (that is, Kolmogorov) complexity. Note that, at least from the perspective of Zurek (1989), this resource arises from information-theoretic (and, specifically, entropy-related) concerns; accordingly, (reversible) computation of an injective function is deemed to have negligible thermodynamic cost, regardless of (for example) the energy inefficiency of a physical instantiation of the computation.

We recall (from Lloyd and Pagels 1988) in passing the notion of *thermodynamic depth*, which provides a measure of the amount of information lost in formation of an object (such as a computer) rather than in a computational process. This is not, then, a commodity resource, but rather a manufacturing cost; see Sect. 3.1.

2.4 Model-independent features of resources

Apart from consideration of *resources* (time, space, material/thermodynamic costs, etc.) that are applicable in the context of arbitrary computational models, we may consider *features* that (unspecified) such resources should possess; we briefly discuss two such features now.

2.4.1 Blum's axioms

These two axioms, introduced in Blum (1967), say of a resource that (1) the resource is defined if and only if computations during which the resource is consumed are themselves defined (this is the case, for example, with the Turing-machine resource of *time*: the number of time steps is a well-defined, finite natural number if and only if the computation halts), and (2) it is a decidable problem to check purported measurements of the resource's amount (again, time in the context of Turing machines satisfies this: given a Turing machine, an input value and a purported number n of time steps, we can check—simply by running the computation for n steps and checking for termination at that point and not before—whether the computation really does use n steps).

The axioms are, for our purposes, desirable,[9] and we stipulate that commodity resources satisfy them. However, necessary as we deem the axioms to be, they are not *sufficient*; we note in Blakey (2008a) that a notion of commodity resource constrained only by Blum's axioms leads to undesirable and deceptive complexity behaviour[10]—whereas tools exist (see Blakey 2008a, b) to determine which of several resources are 'relevant' to a given computation, these tools fail when our concept of resource is not restricted further than by the axioms.

One restriction to the definition of resource that mitigates this deceptive complexity behaviour is to stipulate that resources be *normal*; we now briefly describe the features of normalization of which we make use here, deferring a full account to Blakey (2008a).

2.4.2 Normalization

Roughly speaking, a *normal* resource is one that attains all natural-number values: a resource is normal if and only if, for any natural number n, there exist a computer Φ and an input value x such that Φ, in processing x, consumes exactly n units of the resource. *Normalization* is a process whereby non-normal resources can be converted into normal resources that are order-isomorphic with the originals.[11]

If we were to allow as a resource an *arbitrary* function with codomain $\mathbb{N} \cup \{\infty\}$, then the resource effectively returns 'cardinals': the resource *counts* time steps, tape cells or similar, and we have an intrinsic *unit of measurement*. This seems resource-dependent and not conducive to comparison (for example, how many time steps should we deem of equivalent cost to one tape cell?). If, on the other hand, we allow only *normal* resources, then we have 'ordinals', with 0 representing the least possible resource consumption, 1 the second-least, 2 the third-least, etc.; this is independent of the choice of resource, and of any unit of measurement suggested thereby, and so fairer, resource-heterogenous comparison becomes available.

If we stipulate that our commodity resources be as described in Sect. 2.2, satisfy Blum's axioms and be normal, then (as hinted at above and discussed in Blakey (2008a)) we find that much of the deceptive complexity behaviour alluded to above—notably, that of Footnote 10—is avoided.

2.4.3 Summary of 'Resource'

We summarize now our restrictions on the notion of resource. In the present work, a valid (commodity) resource

– is a function, dependent upon the choice of computer, that maps input values to natural numbers (or to ∞) (see Sect. 2.2);

[9] In particular, we do not consider here *non-deterministic* (commodity) resources, in which case we should not want axiom (2) to hold.

[10] Specifically, we demonstrate in Blakey (2008a) that the importance of certain resources can be artificially exaggerated, as follows. We may, for example, count a Turing-machine computation's *time steps* (let T be their number) and *tape cells* (S); then our measure T of time is more 'significant' than that, S, of space in that $T \geq S$. However, we may (perversely but perfectly validly) measure space instead as 2^S (with the mapping $k \mapsto 2^k$ establishing an order-isomorphism between the two spatial measures, demonstrating their 'equivalence' in some sense), whence it is for some Turing machines the case that space appears more significant than time (in that $2^S > T$). It is this undesirable freedom to engineer resources' apparent relative significance that motivates the below-described restriction to *normal* resources.

[11] The resource S of Footnote 10, then, is normal, whereas 2^S—which normalizes to S—is not.

– satisfies Blum's axioms (see Sect. 2.4.1); and
– is normal (see Sect. 2.4.2).

These are necessary conditions for a resource to be 'valid', though are not between them sufficient[12]; a full *definition* of resource remains an open problem, which is to be discussed further in the wider project of which the present paper is part.

2.5 Resource as a lower bound

We comment in passing that specific values of resources and of complexity functions are viewed as *lower bounds* on what is needed for a computation to succeed—recall (for example from Footnote 4 of Blakey (2008b)) the assumption that a computation can still proceed with more resource than is necessary (a desirable byproduct of this is that our unconventional-complexity definitions are in some respects analogous to their traditional counterparts).[13] However, note that, in a quantum (and, hence, quantum-computing) context, additional 'possibilities' (for example, potential routes taken by photons in the double-slit experiment) may interfere with *and cancel out* existing ones (see Penrose 1999); such phenomena should be considered, then, when selecting resources, so that provision of extra resource cannot, all else being equal, preclude a computation.

2.6 Specific, model-dependent resources

We now consider specific (illustrative rather than exhaustive) models of computation, and ask which resources are likely to be relevant for instances thereof.

2.6.1 Actual, physical implementations of Turing machines

We have already commented that, for the abstract, unimplemented Turing-machine model itself, the resources of time and space (and variants thereof) are sufficient for complexity-theoretic purposes. The resource of *precision*, then, is not a direct concern for Turing machines:

> [w]hat is fundamental about the idea of a Turing Machine and digital computation in general, is that there is a perfect correspondence between the mathematical model and what happens in a reasonable working machine. Being definitely in one of two states is easily arranged in practice, and the operation of real digital computers can be (and usually is) made very reliable. (Vergis et al. 1986)

[12] An illustration of this insufficiency arises from the fact that, though normalization precludes *exaggeration* of a resource's importance via application of functions such as $k \mapsto 2^k$, it does nothing to preclude *understatement* of a resource's importance via application of functions such as $k \mapsto \lfloor \ln k \rfloor$.

[13] We must clarify this point: the sense in which resource and complexity are *lower* bounds is that a computation *with fixed input value* may proceed with at least these bounds' allocation of resource; extra resource beyond that prescribed by a resource/complexity function is not problematic. This is in contrast with, and should not be confused with, the observation that a complexity function is the *maximum* (over input values of a certain size) amount of resource sufficient for a computation to proceed, in which sense complexity functions are *upper* bounds.

However, both the *alphabet size* and *number of states* relate to precision in that distinguishing more distinct symbols entails smaller differences therebetween, resulting in smaller differences (in voltage or similar) between their respective real-world implementations.[14]

Nonetheless, the alphabet case is not problematic: 'meta-symbols' comprising several symbols can be used instead of several individual symbols; similarly, the states may be encoded via such 'place notation'. Further, the Turing machine's unbounded tape seems unproblematic, as long as by 'computer' we mean not a fixed-memory machine, but the machine plus arbitrary additional memory.

This suggests that consideration of time and space alone may still be enough, and (unsurprisingly) Turing (1936) argues similarly (noting in particular the issue of alphabet size/symbol differentiation). However, from a *complexity* point of view, though real-world computers may offer a good, finite approximation of Turing machines, we are interested in *asymptotic* behaviour of resources, and so precision should be accounted for.

So, we consider the resources of *time* and *space* (which latter accounts for the unboundedness of a Turing machine's tape: the resource quantifies how much space is needed given a certain input size, and may then determine adherence or otherwise to financially/technologically/geographically imposed bounds on the space available to a physically implemented computer) as normal, and add to these *precision* so as to account for symbol and state numbers (and, hence, indirectly, for the size of the Turing machine's transition table). These are the only significant differences between a Turing machine and a real-world implementation as far as complexity resources are concerned.

2.6.2 *Analogue/kinematic computers*

Time and space need, as always, to be considered when working with this model. However, they alone are not sufficient: recall the factorization system of Sect. 1.3. One additional relevant resource, as we have seen, is *precision* (this is also evident from, for example, the greatest common divisor system described in Blakey (2007b), the wedge-detection cannon system of Beggs et al. (2008) and the Differential Analyzer of Bush (1931)), and, we see below, there are others as well.

It seems intuitively clear that we should consider also the *energy* required to drive the computer. (Energy was not considered in the case above (namely, real-world implementations of Turing machines) since, assuming 'ballistic' computation where the processor is used at capacity without, in particular, pauses for user interaction, energy consumption is linear in run-time and therefore redundant from a complexity-theoretic perspective (provided that we have not neglected to consider time).

In Vergis et al. (1986), the resource of precision is dealt with by acknowledging that an analogue computer has some 'ε' of imprecision, which value is fixed a priori and determines the maximum size of input that can be processed successfully, therefore not rendering precision a resource in our (commodity) sense. However, various non-standard resources (in our commodity sense)—physical size, mass, initial stored energy, time interval of operation—are considered, though these reduce to time and space (due to bounded density, etc.).

[14] Strictly speaking, symbol and state numbers are a priori fixed, whereas we as complexity theorists should like to think in terms of functions of input size; accordingly, we may consider measures such as 'number of distinct symbols/states *used so far*' as a function of input size.

In summary, then, relevant resources for analogue/kinematic computers include *time*, *space*, *precision* and *energy*.

2.6.3 Relativistic computers

The broad idea of relativistic computation (of which the detailed physics is beyond the scope of the present paper) is to exploit relativistic effects that allow a computer to experience time at a greater rate than its user; for then the user need wait less time (than if the user's and computer's clocks agreed) for an output value: more (computer) time steps are accommodated in each (user) second. Suggestions of how to achieve this effect include sending computers through wormholes, etc.; the situation is sometimes contrived such that an *infinite* amount of computer time elapses in a finite amount of user time, whence hypercomputation becomes available.

When considering resources for this model, there are two points to consider.

First, the resource of *time* is now ambiguous: we may consider seconds (say) counted by the *computer* or by the *user*. The tacit assumption is that the latter reference frame is more important, for else no computational speed-up is achieved. This assumption is perfectly reasonable: we view the computer, wormhole, etc. (but not the user) *together* as the computing system, and measure the time for which the user has to wait—this seems the most natural choice to resolve the ambiguity of the resource of time, and the most natural generalization of the resource as in other models.

Secondly, note that, for present purposes, we adopt a fairly practical stance when considering models of computation: when identifying resources, for instance, we are careful to distinguish between (abstract) Turing machines and their (physical) implementations; and we consider practical issues such as achievable precision. In this context, then, it seems reasonable to exclude relativistic computers (at least as a form of hypercomputer) on the grounds of (for example) their energy consumption: although the user experiences only a finite amount of time, the computer needs to be *powered* for what the computer itself deems an eternity—we use relativity as an attempt to bypass *time* restrictions, but other resources' constraints (energy, durability of the physical machine, etc.) are still present. We see in another guise our original contention: time (and space) are not the only complexity-theoretic resources, and should not be treated as such. (There may, prima facie, still be an advantage offered by relativistic computers, in particular where computations are time-heavy; however, maintaining a computer's running appears to require other resources (energy or similar) in proportion to time, which suggests that a computation is never truly (uniquely) time-heavy.)

The power of this paradigm, then, comes primarily from neglect of the computer's time-frame in favour of the user's, though the availability (to the computer) of other resources such as energy is still an issue.

(Commodity resources of user- and computer-time aside, there is clearly a significant *non*-commodity—specifically manufacturing—cost incurred during production of the wormholes, black holes, etc. used by this computational paradigm; see Sect. 3.1.)

2.6.4 Optical computers

Woods (2005) introduces an optical computer that computes via image manipulation; the resources considered are: time; the number of grid images; spatial, amplitude and phase resolutions; dynamic range; and frequency of illumination. Without justification here, we state that these resources are akin to forms of *time*, *space* and *precision*, of which each is

relevant for optical computers. We also suggest that *energy* is a relevant resource to the wider paradigm of optical computers, since some instances rely, for example, on the availability of electromagnetic waves of a prescribed *wavelength* (which may depend on the input size).

In summary, then, relevant resources for optical computers are *time*, *space*, *precision* and *energy* (and variants thereof).

2.6.5 Quantum computers

We defer to future work a detailed account of the computational resources consumed by quantum computers, but make some general comments here.

In *circuit-model* quantum computation (wherein input is encoded via preparation of several quantum bits, processing takes the form of the application of unitary operations to subsets of these quantum bits, and output is via measurement of the system), we note that a commonly used complexity measure is the *number of invocations of unitary operations* (Nielsen and Chuang 2000). This measure essentially captures the system's *run-time* (just as a Turing machine's run-time can be defined as the number of invocations of atomic, one-time-step operations whereby tape cell, machine state and head position are updated), which is not, we suggest, a particularly insightful measure for quantum computers: the benefit enjoyed by quantum computers over their classical counterparts is gleaned primarily from the use of entangled states, and the effective parallelism that this allows; a drawback is the strictly constrained way in which measurements can be taken of the system; run-time, then, is a reflection of neither the 'amount of computation' performed (since parallelism is not taken into account) nor the 'difficulty' in using the system (notably during measurement).

We now consider the *adiabatic* quantum model (where output values are encoded in the final ground state of an evolving system, with the evolution proceeding from an achievable initial ground state sufficiently slowly that no higher energy state is reached, so that our 'output-value' final ground state is indeed encountered). Standard expositions of the paradigm consider *time* as the only resource, as, indeed, is tacit in our "sufficiently slowly" above; however, determining this sufficient time makes use of trade-offs with other resources. For example, the time sufficient for an evolution to remain in the ground state is a function of the minimum gap between the 0th (ground) and 1st energy states; *energy*, then, is an important (though popularly 'behind-the-scenes') resource for this computational model.

The *measurement-based* quantum computation model sees a computation take the form of several measurements (which can, in principle, be performed simultaneously), each of an individual quantum bit from an initial, large, entangled (cluster) state. As this description suggests, the complexity resources relevant to such a computation may be markedly different from those used in the circuit and other quantum models: enumerating invocations of unitary operations no longer applies in a straightforward manner; in fact, a relevant view of (non-commodity) 'complexity resource' in this context concerns the difficulty in producing the initial cluster state—see Sect. 3.1.1.

2.6.6 Chemical computers

We mention here one specific resource that is particularly relevant to chemical computers, namely *mass*. We recall from Adleman (1994) that DNA computers offer an approach to the (NP-complete) Travelling Salesperson Problem, and that the time (and, for that matter,

energy) complexity of this method appears acceptable. However, as is pointed out in Hartmanis (1995), the *mass* of DNA required by the method in processing non-trivial problem instances is greater than the mass of the Earth![15] We recall the long-held, de facto rule of thumb that tractability corresponds to polynomial resource consumption, and note that, in this case, the resource of *mass* imposes an exponential cost, and, therefore, intractability.[16]

(As an aside, note that, due to the bounded density of chemical-computing apparatus—including DNA itself—, mass is bounded by a constant multiple of space, and so the resource of mass tells us little new, provided that we consider space. However, the distinction is illustrative of the unexpected ways in which complexity (in this example space complexity) can be affected by strictly unconventional-computing concerns.)

3 Other resources

We now mention in passing some (but by no means all) *non-commodity* interpretations of 'resource' (though justify briefly our focus[17] on commodity resources by citing (a) analogy with traditional complexity classes, where non-commodity resources such as the computational use of non-determinism are accounted for in the classes themselves; and (b) the fact that many non-commodity resources can be viewed as features of the computational model, which is accommodated by our model-independent approach to complexity theory).

By 'resource', we no longer necessarily mean 'commodity resource'.

3.1 Non-commodity resource types

3.1.1 Manufacturing costs

We may view as a resource the costs of *constructing* (rather than running) a computer. These may, for example, include the cost (whether this be financial, thermodynamic—see Lloyd and Pagels (1988)[18] and Zurek (1989)—, etc.) of manufacturing the system's physical structure (including the structure of wormholes, black holes, etc. in the case of relativistic computing), or of producing an entangled state to be used in a quantum computation (e.g., a cluster state from which measurement-based quantum computation proceeds).

Viewing living brains as computers, we may also reasonably include under this resource such measures as time taken to evolve.

[15] Hartmanis (1995) writes of *weight*, but strictly means *mass*; the distinction is important since we are dealing with masses of the order of that of Earth, whence we may no longer assume negligible changes in gravitational strength from one part of the computational apparatus to another.

[16] An alternative view of mass in this instance is as a measure of the number of parallel 'processors' at work during a chemical computation: the exponential speed-up observed with the TSP system and similar stems essentially from the presence of exponentially many DNA strands simultaneously testing one potential solution each. Recall, however, the discussion of parallelism in Sect. 2.1.

[17] Because of this focus, a more rigorous description of 'non-commodity resource' than is presented here is not necessary; furthermore, such description is rendered elusive by (amongst others) issues described in Sect. 3.2.

[18] We recall from Lloyd and Pagels (1988) the complexity measure of *thermodynamic depth*, which gauges the loss of information during formation of an object (e.g., a computer). In relation to this measure, and in agreement with our general thesis that unconventional computers warrant unconventional resources, Lloyd and Pagels (1988) write that, "if a definition of complexity is to be a useful measure for physical systems then it must be defined as a function of physical quantities, which in turn obey physical laws."

3.1.2 Features of computational models

Computation may be facilitated by taking advantage of 'permissions' granted by the computational model. For example, we may augment a Turing machine by allowing it to use *non-determinism*; then, although the same problems are *computable*, we do at least enjoy an apparent (and, if $\mathsf{P} \neq \mathsf{NP}$, a definite) increase in (time) *efficiency*. Similarly, a computer may be augmented by allowing consultation of *oracles*.

Such permissions are non-commodity resources on which we may draw to aid computation.

3.1.3 Features of enveloping physical laws

Similarly to the features of the *computational model* (see above), we may exploit features of the *physics* describing the model. For example, a quantum computation may rely on the presence of *entanglement*, which entails our adopting a non-classical physics, whereas, when using other computation models (such as centimetre-scale kinematic computers), it may be convenient to assume that entanglement (and other non-classical phenomena) are *not* present, and that we may safely assume Newtonian dynamics.

Again, such permissions are non-commodity resources on which we may draw to aid computation (and to aid our describing and reasoning about computation).

(We mention in passing Heisenberg's uncertainty principle, to which are closely related trade-offs between precision and other (commodity) resources such as energy. Further discussion is beyond the scope of the present paper.)

3.1.4 Information-theoretic resources

Another class of resources that are non-commodity, but on which we can nonetheless draw for computational gain, is the class of 'information-theoretic' resources. These include (classical and quantum) *communication channels*, *entangled states*, etc.

We may, for example, summarize the information-theoretic content of the quantum teleportation protocol (Bennett et al. 1993) with the inequality,

$$2\textit{cl-bit} + \textit{e-bit} \geq \textit{qubit},$$

where '*cl-bit*' stands for the resource of being able to transfer a classical bit, '*qubit*' stands for the resource of being able to transfer a quantum bit, '*e-bit*' stands for the resource of having available an (entangled) Bell state, and '$X \geq Y$' indicates that resources X are at least as powerful as (i.e., they can simulate) resources Y.

3.2 Recasting as different resource types

Note that the boundaries between 'types' of resource (commodity, model-feature, physics-feature, information-theoretic, etc.) are not always clear: there is not always a unique 'type' to which a given computational resource belongs. For example, the resource of *non-determinism* is discussed above in the context of being a model-feature resource; however, its presence or otherwise could equally be considered a physics-feature resource (where the presence or otherwise in physics of pre-ordainment is the determining factor), or even a commodity resource (where we may consider the number of non-deterministic bits, say, consulted during a computation).

However, the crucial point is not that resources fall neatly into these categories, but that practitioners of unconventional computing consider all resources (from whatever categories they may be) relevant to their systems.

4 Summary

We reiterate our main claim: that successful analysis of the complexity of *unconventional* computers requires consideration of *unconventional* resources. This claim is justified by the motivating example of the factorization system, the true complexity of which conventional resources alone fail to capture. We discuss above various non-standard (commodity) resources, which are relevant to various computational models; we discuss also different, non-commodity interpretations of 'resource'.

We hope that the issues raised here provoke thought amongst the unconventional-computing and computational-complexity communities, and lead to more rigorous and complete analyses of unconventional systems' computational complexity.

Acknowledgments We thank José Félix Costa for his kind invitation to contribute this paper; Bob Coecke and Joël Ouaknine for their continued support and supervision; and conference participants, collaborators and colleagues who have helped to shape and direct this research. We thank the anonymous reviewers of an earlier version of this paper for their detailed and useful suggestions. We acknowledge the generous support of the EPSRC; this work forms part of the EPSRC-funded project *Complexity and Decidability in Unconventional Computational Models* (EP/G003017/1).

References

Aaronson S (2005) Complexity Zoo. Online resource, available at http://www.qwiki.stanford.edu/wiki/Complexity_Zoo

Adleman L (1994) Molecular computation of solutions to combinatorial problems. Science 266:1021–1024

Beggs E, Costa J, Loff B, Tucker J (2008) The complexity of measurement in classical physics. In: TAMC 2008, LNCS 4978

Bennett C (1982) The thermodynamics of computation—a review. Int J Theor Phys 21(12):905–940

Bennett C, Brassard G, Crépeau C, Jozsa R, Peres A, Wootters W (1993) Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. Phys Rev Lett 70:1895–1899

Blakey E (2007a) An analogue solution to the problem of factorization. Oxford University Computing Laboratory Research reports series, RR-07-04

Blakey E (2007b) On the computational complexity of physical computing systems. In: Adamatzky A, Bull L, De Lacy Costello B, Stepney S, Teuscher C (eds) Unconventional computing 2007. Luniver Press, Beckington

Blakey E (2008a) Beyond Blum: what is a resource? Int J Unconv Comput 6(3–4):223–238

Blakey E (2008b) Dominance: consistently comparing computational complexity. Oxford University Computing Laboratory Research reports series, RR-08-09

Blakey E (2008c) Factorizing RSA keys, an improved analogue solution. New Gener Comput 27(2):159–176

Blakey E, co-prepared by patent attorneys (2008d) System and method for finding integer solutions. United States patent 7453574

Blum M (1967) A machine-independent theory of the complexity of recursive functions. J Assoc Comput Mach 14(2):322–336

Blum L, Cucker F, Shub M, Smale S (1997) Complexity and real computation. Springer, New York

Brent R (2000) Recent progress and prospects for integer factorisation algorithms. LNCS 1858. Springer-Verlag, Berlin

Bush V (1931) The Differential Analyzer: a new machine for solving differential equations. J Franklin Inst 212:447–488

Hartmanis J (1995) On the weight of computations. Bull EATCS 55:136–138

Landauer R (1961) Irreversibility and heat generation in the computing process. IBM J Res Dev 5(3):183–191
Lloyd S, Pagels H (1988) Complexity as thermodynamic depth. Ann Phys 188:186–213
Nielsen M, Chuang L (2000) Quantum computation and quantum information. Cambridge University Press, Cambridge
Păun G (2008) From cells to (silicon) computers, and back. In: Cooper B, Löwe B, Sorbi A (eds) New computational paradigms; changing conceptions of what is computable. Springer, Heidelberg
Penrose R (1999) The emperor's new mind: concerning computers, minds, and the laws of physics, 2nd edn. Oxford University Press, USA
'Resource' dictionary entry (1989) Oxford English Dictionary, 2nd edn. Oxford University Press
Turing A (1936) On computable numbers, with an application to the Entscheidungsproblem. Proc Lond Math Soc 2(42):230–265
Vergis A, Steiglitz K, Dickinson B (1986) The complexity of analog computation. Math Comput Simul 28(2):91–113
Vitányi P (2005) Time, space, and energy in reversible computing. In: Proceedings of the 2nd conference on Computing frontiers
Woods D (2005) Computational complexity of an optical model of computation. PhD Thesis, NUI Maynooth
Zurek W (1989) Thermodynamic cost of computation, algorithmic complexity and the information metric. Nature 341:119–124