Beyond Blum: What is a Resource?

ED BLAKEY*

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

Received on 23 September, 2008, and, in final form, on 20 November, 2008.

When analysing a *Turing machine's* complexity, we can appeal to decades of experience to determine which resources (typically time steps or tape cells) to measure. More fundamentally, we have Blum's criteria for admission as a valid resource.

When analysing a *non-Turing* computer's complexity, the situation is less clear. What resources are relevant for, say, an analogue computer? Can we meaningfully compare a Turing machine's time complexity with an optical computer's precision complexity? Crucially, *what should we admit as a resource* in the context of non-standard computation?

Our aim is to specify a suitable, non-Turing-computer (in fact, not-necessarily-Turing-computer) analogue of Blum's axioms. We start with the existing axioms, but show that, alone, they are insufficient. Accordingly, we further restrict the notion of resource: we define *normality* of resource, advocating consideration only of *normal* resources during non-standard computers' complexity analyses. This eliminates some 'deceptive' complexity behaviour encountered with Blum's axioms alone.

Key words: Computational complexity; non-Turing/non-standard computer; resource; Blum's axioms; normalization; dominance.

* email: edward.blakey@queens.ox.ac.uk

1 INTRODUCTION

The purpose of analysing computational complexity is to measure the *cost* of performing a computation; this cost is the amount of *resource* consumed during the computation. Of particular interest to the complexity theorist is the *scaling behaviour* of these amounts of resource, i.e., the asymptotic $\mathcal{O}$-behaviour of the amounts viewed as functions of the size of the computation's input.

Devices performing computations of which the complexity is so analysed are typically modelled as *Turing machines* (or, equivalently, as instances of some other algorithmic computational model: random access machines, finite state automata, programs written in a suitable language, etc.). Accordingly, the resources most commonly considered are those—run-time, memory space (e.g., tape cells), and, less frequently, ink, head reversals, etc.—tailored specifically to and catering specifically for such machines (see, for example, [7] and [9]).

Bearing in mind the growing recognition (see [1], [8], etc.) of the importance of *non-Turing* computation, we consider in [2] ways in which to analyse the complexity of non-standard (quantum, chemical, analogue, optical, etc.) computers, advocating in particular the consideration not only of algorithmic, 'Turing-machine-type' resources (run-time, memory space, etc.), but also of non-algorithmic, 'physical-computer-type' resources (precision, energy, etc.*). The message is that

> *standard complexity analysis sometimes overlooks the true complexity of non-standard computers because the correct resource types are not considered*.†

The problem readily suggests its own solution, namely consideration of the correct resource types; but *what is a resource type*? This is the question under

* None of these 'physical' resources has been studied particularly uniformly; this will to some degree be redressed during the progress of the wider project of which the current work is part.

† We discuss elsewhere an illustrative example of such overlooking: in [2] and [3], we describe an analogue factorization system; we note in [3] that an exclusively 'Turing-type-resource' analysis suggests that the system consumes resource only *polynomial* in the size of the input value—notably, time and space complexities are polynomial—, whereas the system's true, *exponential* complexity is only apparent when 'non-Turing-type' resources such as *precision* are considered. Intuition about the general phenomenon at play here can be gleaned by noting that, if input/output values are encoded in the precise details of, say, the positions of a system's constituent parts, and if processing is via typically parallel physical evolution, then complexity measures such as time and space may be less relevant—in that they may reflect less accurately the *amount of computation* being performed—than resources such as precision that are not applicable to Turing machines.

consideration here. (The question—together with the germ of an approach—was raised in footnote 35 of [5], which paper inspired the proposed project of [2], of which the present work is part.)

We stress that we have in mind no specific computational model, since we wish to retain generality and broad applicability. We require of a computing system only that it have provision for accepting input values (encoded in the values of manipulable parameters), for processing such (via the system's evolving in accordance with the laws—be they electrical, chemical, quantum-mechanical, etc.—to which it is subject), and for supplying corresponding output values (encoded in the values of measurable parameters).[‡] Note that this view of computation accommodates, amongst many other things, physical implementations of (finite-tape) Turing machines, such as the digital computer. Ultimately, we are interested here in the properties of *resources*[¶] themselves, regardless of the computational models from which are taken the computers that consume these resources.

1.1 Introducing Resource

As in [4], we view *resources* (denoted A, B, C, etc.) as functions

- that depend on the choice of *computational system* (shown as a subscript, which may, when understood, be suppressed), and
- that map an *input value* to the corresponding amount of *resource* used by the system in processing this value.

Hence, where Φ is a computer and x an input value for Φ, $A_\Phi(x)$ (or simply $A(x)$) is the amount of resource A consumed by Φ in processing x. More specifically, $A(x)$ is the (integer) *number of units* of A consumed, since we stipulate that A have codomain $\mathbb{N} := \{0, 1, 2, \ldots\}$ (we implicitly use this fact in the proof of Theorem 1, for example); should our resource instead lend

[‡] We tacitly assume here that the computation will end after only finite time (so that the output values become available). Note, however, that certain computational models that do not satisfy this criterion can nonetheless be accommodated: *interactive* computation, for example, can be divided (at points of user interaction) into subcomputations of the type stipulated. Further, it is possible to extend our interpretation of computation to include genuinely infinite computations such as those encountered with *relativistic* computing, etc. (we intend to address this issue in future work), thereby accommodating yet more paradigms.

[¶] Of course, there are many important aspects of computational complexity theory besides the notion of resource; our exclusive focus here on resource reflects only the fact that this work is part of a wider project, of which other aspects are investigated elsewhere (for example, in [2] and [4] we discuss the way in which, given an understanding of resources, we automatically obtain corresponding *complexity classes*).

itself to a real-number, continuous codomain (as does energy, for example, at least at super-quantum scales), then we may apply a rounding function (say, $x \mapsto \lceil x \rceil$) to restore a natural-number codomain. This loses no relevant information: we, as complexity theorists, are interested in the asymptotic scaling behaviour of resources as encapsulated by $\mathcal{O}$-notation, which notation 'smooths over' fine details such as the effects of our rounding function.

Note as an aside that this description of resource does not accommodate everything that can, intuitively, be considered a resource: when we talk of 'resource', we mean resource as *consumed* by instances of whichever computational model is under consideration. This model a priori allows non-determinism, or a priori does not (depending on the model); similarly probabilism, use of oracles, etc. Whilst non-determinism, probabilism, etc. are (intuitively) resources on which we may draw in order to augment computing power, then, and whilst they are accommodated by our view of computation (since we do not specify a paradigm), they are not modelled by what we here call 'resources'.

Resources have associated *complexity functions*, which we now define. Given a computational system, we may ask how a specific resource scales. In particular, we may be interested not in the resource consumed by the system given *one specific input value* (that is, in some '$A(x)$'), but in the resource consumed *as a function of the size of the input value*[§] —this is what we mean by the *complexity function* corresponding to the resource. Specifically, we have the following.

Definition 1 (complexity function). Let Φ be a computer with set X_Φ of input values, let σ be a size function, and let A be a resource. The *complexity function* $AC_{\Phi,\sigma}$, corresponding to resource A, is given by

$$AC_{\Phi,\sigma}(n) := \sup\{A_\Phi(x) \mid x \in X_\Phi \wedge \sigma(x) = n\}$$

('Φ's and/or 'σ's are often suppressed when understood).

[§] We require, then, a *size function* that maps input values to non-negative reals (the input values' *sizes*); for example, if our input value is a natural number n expressed in binary, then we can take as its size the number of bits, excluding leading zeros (or, as is sufficient for virtually all complexity-theoretic purposes, the approximation $\log_2(n)$ to this number of bits). Size functions for *real-number* input values may, depending on application and the values' encoding, map x to $\log x$, to the combined number of digits and decimal places of x, to the Kolmogorov (algorithmic) complexity of x, to the combined number of digits of the numerator and denominator of a (fully cancelled) rational expression of x, or similar.

So, just as A, B, C, etc. stand for types of *resource*, AC, BC, CC, etc. stand for types of *complexity*.

Note that we have not *defined* resource. The description given here of resource is necessary for our purposes, but not sufficient; further restriction, we see below, is required. (We have, however, defined complexity in terms of resource.)

2 RESTRICTING RESOURCE

As we point out in [2], when analysing the complexity of a given (possibly non-standard, non-Turing) computing device, we need to know which resources to consider; we stress this in [2] and [3] by showing that failure to consider all appropriate resources can give an unrealistically optimistic impression of a system's complexity (with supposedly polynomial-resource factorization being the illustrative example of these two cited works). There are two approaches to this issue of identifying appropriate resources, one practical and ad hoc, the other (on which we focus here) theoretical and abstract; the latter approach sees us ask, regardless of choice of computing device, *what should we admit as a resource*?

- PRACTICAL APPROACH. With a specific computing device in mind, we may consider in an ad hoc way its implementation, looking for bottlenecks and impracticalities and restating such as resource constraints. For example, the result of a computation by some analogue system, say, may be encoded in a real-number position coordinate of some part of the system—the computation tacitly relies, then, on our being able to achieve infinitely precise measurement in order to retrieve this real number—, whereas in practice our measured value may be known only to be within some $\epsilon > 0$ of the true value; this prompts us to consider the resource of *precision*, and we may for example find that, as the input value grows, the precision required during measurement increases exponentially.[||]

- THEORETICAL APPROACH. We may question, regardless of computing system, what a resource can (and should) be. Blum's axioms—see Sect. 2.1—offer a starting point. With the wider complexity framework of [2] in mind, we may restrict the notion of resource so as to be compatible with comparisons made with respect to *dominance* (whilst we

[||] The interested reader is invited to compare this situation with that of the analogue factorization system of [2], where precision transpires to be paramount.

define and discuss dominance in [4], detailing the according comparison of resources, we also reproduce in Sect. 2.2 details that are necessary/useful for present purposes); to this end, we define and consider below resource functions' *normality*—see Sect. 3.

Little can be said, at least in generality, about the former, practical approach due to its ad hoc nature. Accordingly, we focus on and develop the latter, theoretical approach here.

2.1 Blum's Axioms

In standard complexity theory, i.e., complexity theory as it applies to Turing machines, Blum's axioms (introduced in [6]) may be—and typically are (although exceptions occur during consideration of non-deterministic resources, for example)—used to constrain what is allowed to be a resource. (More generally than Turing machines, the axioms' original scope—see [6]—is *algorithmic computers*, of which our area of interest is nonetheless a strict superset.) The axioms ensure

1. that a measure of resource is defined precisely at inputs at which the computation being measured is defined, and

2. that it is a [Turing-] decidable problem to determine whether a given value is indeed the measure of resource corresponding to a given input.

Blum's axioms may seem a good starting point for our attempt to restrict suitably the notion of resource in the context of non-standard computing. Axiom 1 is automatically satisfied (this is implicit in the second bullet point of the description of resource in Sect. 1.1, wherein the implication is that an undefined computation, and only an undefined computation, gives rise to an undefined amount of resource), and—for our measures of resource to be of any practical value—the necessity of axiom 2 is clear. Further, our adoption of the axioms gives us many standard results from [6]; e.g., (Theorem 3 of [6]) for resources A and B, and for dummy variable x standing for an input value, there exists a computable function g such that $g\left(x, A\left(x\right)\right) \geq B\left(x\right)$ and $g\left(x, B\left(x\right)\right) \geq A\left(x\right)$ with only finitely many exceptions x.

However, the axioms alone are not enough: we note below a problem that they do nothing to solve (see Sect. 2.2: *The Problem: Dominance's Incompatibility with Unrestricted Resource*); preliminarily to the discussion of the problem, we now introduce *dominance*.

2.2 Dominance

Whilst consideration of many different resources (both algorithmic—time, space, etc.—and non-algorithmic—precision, energy, etc.) facilitates more insightful analysis of computers' (especially non-Turing computers') complexity, it complicates the issue of *comparison* of computers' respective complexity functions, simply because there are more candidate functions potentially to compare. We can compare time complexity (of one computer) with time complexity (of another), precision with precision, and so on, but it is unclear which, if any, of these comparisons are meaningful.

The notion of *dominance* (see [4] for a more complete account thereof) was introduced as a means of formalizing resources' *relevance*; this determines between which resources' complexity functions meaningful comparisons may be made: we can determine relative efficiency by comparing (according to the '$\in \mathcal{O}$' pre-ordering of functions) computers' respective consumptions of *relevant* resources.

We stress that the intention of dominance is to allow comparison between *computers*, i.e., between *instances* of (possibly different) computational models, rather than between the models themselves; this is in contrast with the approach of Udi Boker and Nachum Dershowitz, for example.

We now recap from [4] the definition of $\mathcal{R}$-dominant resource, and introduce $\mathcal{R}$-complexity.

Definition 2 ($\mathcal{R}$-dominance; $\mathcal{R}$-complexity). Let Φ be a computing system, and let $\mathcal{R}$ be a finite, non-empty set of resource types for Φ.

- An *$\mathcal{R}$-dominant resource* for Φ is a resource $A \in \mathcal{R}$ such that, for any resource $B \in \mathcal{R}$ such that $AC_\Phi \in \mathcal{O}(BC_\Phi)$, we have that $BC_\Phi \in \mathcal{O}(AC_\Phi)$.

  (Thus, A is $\mathcal{R}$-dominant if and only if each $B \in \mathcal{R}$ is either

  - $\mathcal{O}$-incomparable with A ($AC \notin \mathcal{O}(BC)$ and $BC \notin \mathcal{O}(AC)$),
  - $\mathcal{O}$-dominated by A ($BC \in \mathcal{O}(AC)$ but $AC \notin \mathcal{O}(BC)$), or
  - $\mathcal{O}$-equivalent to A ($AC \in \mathcal{O}(BC)$ and $BC \in \mathcal{O}(AC)$);

  that is, no $B \in \mathcal{R}$ $\mathcal{O}$-dominates A (whence we would have that $AC \in \mathcal{O}(BC)$ but $BC \notin \mathcal{O}(AC)$).)

- The *$\mathcal{R}$-complexity* of Φ (for size function σ), denoted $\mathcal{B}_{\mathcal{R},\Phi,\sigma}$ (from which we may omit any combination of subscripts when understood),

is the complexity function given by

$$\mathcal{B}_{\mathcal{R},\Phi,\sigma}(n) := \sum_{R \text{ is } \mathcal{R}\text{-dominant}} RC_{\Phi,\sigma}(n) \quad .$$

The intent of this definition is that comparisons can be made between computers' respective *relevant* resources, as encompassed by $\mathcal{R}$-complexity. Accordingly, we make the following definition.

Definition 3 (greater/equal efficiency; lower/equal complexity; incomparability). Relative to $\mathcal{R}$ and σ, we say of computers Φ and Ψ that

- Φ is *more efficient* (has *lower complexity*) than Ψ, denoted '$\Phi \lesssim_{\mathcal{R},\sigma} \Psi$' (or '$\Phi \lesssim \Psi$' if $\mathcal{R}$ and σ are understood), if $\mathcal{B}_{\mathcal{R},\Phi,\sigma} \in \mathcal{O}(\mathcal{B}_{\mathcal{R},\Psi,\sigma})$ but $\mathcal{B}_{\mathcal{R},\Psi,\sigma} \notin \mathcal{O}(\mathcal{B}_{\mathcal{R},\Phi,\sigma})$;
- Φ and Ψ are *equally efficient* (have *equal complexity*), denoted '$\Phi \simeq_{\mathcal{R},\sigma} \Psi$' (or '$\Phi \simeq \Psi$' if subscripts $\mathcal{R}$ and σ are understood), if both $\mathcal{B}_{\mathcal{R},\Phi,\sigma} \in \mathcal{O}(\mathcal{B}_{\mathcal{R},\Psi,\sigma})$ and $\mathcal{B}_{\mathcal{R},\Psi,\sigma} \in \mathcal{O}(\mathcal{B}_{\mathcal{R},\Phi,\sigma})$; and
- the respective efficiencies/complexities of Φ and Ψ are *incomparable*, denoted '$\Phi \wr_{\mathcal{R},\sigma} \Psi$' (or '$\Phi \wr \Psi$' if $\mathcal{R}$ and σ are understood), if neither $\mathcal{B}_{\mathcal{R},\Phi,\sigma} \in \mathcal{O}(\mathcal{B}_{\mathcal{R},\Psi,\sigma})$ nor $\mathcal{B}_{\mathcal{R},\Psi,\sigma} \in \mathcal{O}(\mathcal{B}_{\mathcal{R},\Phi,\sigma})$.

What we typically compare using this definition is a pair (Φ, Ψ) of computers that solve the same problem, in order to determine which is the more efficient (or, from a more problem-centric point of view, in order to place a tighter bound on the complexity of the problem). The pair may be taken from the same computational model, or from different models (as an example of the latter, we may wish to measure the efficiency of our new unconventional computer against the benchmark of an existing digital computer that solves the same problem). It is, of course, possible to compare the respective efficiencies of computers that solve *different* problems, though the relevance of such comparison is not clear; it is a poor way, in particular, of assessing the relative difficulty of problems—unless we are sure that the compared systems solve their respective problems *optimally*, such assessments are better made via the standard complexity-theoretic notion of *reduction*.

In order to use successfully the notions of Definitions 2 and 3 to compare computers' efficiency, we need to restrict (more stringently than do Blum's axioms) what we allow as a resource, as we now demonstrate.

The Problem: Dominance's Incompatibility with Unrestricted Resource.
Let $S(x) \in \mathbb{N}$ be the number of cells of space used, and $T(x) \in \mathbb{N}$ the number of time steps elapsed, during a computation (by, say, some Turing machine Φ) with input value x.[#] S is, as far as Sect. 1.1 and Blum's axioms are concerned, a legitimate resource, as is S' given by $S'(x) := 2^{S(x)}$; each is an internally consistent measure of space usage, and the mapping $S(x) \mapsto S'(x)$ from $\mathbb{N}$ to $\mathbb{N}$ is isotone (so that ordering input values according to their respective values of S has the same result as ordering by values of S')—we have two seemingly viable, isomorphic (under the isotone mapping) ways of quantifying space usage.

Now if we make inter-resource comparisons according to dominance, then we may find that S' dominates T (more precisely, that S' is $\{S', T\}$-dominant and T is not), but that S does not dominate T (that is, T, but not S, is $\{S, T\}$-dominant)—this is the case, for example, when, for input x of size n, $S(x) \in \mathcal{O}(n)$ (whence $S'(x) \in 2^{\mathcal{O}(n)}$) and $T(x) \in \mathcal{O}\left(n^2\right)$; certain multiplication algorithms, for example, have such complexities.

What we note is that space *measured using* S' appears to make a greater contribution than run-time to the system's complexity, whereas space *measured using* S does not—

> *we can engineer which resource* (in our example, space or run-time) *appears more important,*

and, hence, which contributes (as an addend) to $\mathcal{B}_\Phi$: by applying to the more slowly growing, non-dominant resource a sufficiently fast-growing, monotonic function—in our example, $n \mapsto 2^n$ (more properly, the effective application of this mapping is by way of our choice of $\{S', T\}$, rather than $\{S, T\}$, as resource set)—, this resource becomes dominant.

(Of course, if our notion of resource is sufficiently permissive that *dominance* singles out arbitrary, rather than relevant, resources—as is the case above—, then $\mathcal{R}$*-complexity* and $\lesssim$*-ordering* (both ultimately defined in terms of dominance) suffer a corresponding, knock-on unreliability.)

We now introduce *normalization* as a way of restricting the notion of resource so as to avoid the problem discussed above.

As an aside, note that the problem described here—that the impact of the

[#] Note that our consideration of the resources of *space* and *time* is intended to be illustrative of general, non-resource-specific concepts explored in the present paper. The choice of these particular resources is arbitrary, and we do not rely here or in the related discussion in Sect. 3.1 on results—$S \in \mathcal{O}(T)$, $T \in \mathcal{O}\left(2^S\right)$, etc.—that depend on properties of the specific nature of these resources.

system's space consumption may be *exaggerated* by considering 2^S rather than S—is one side of a coin; the other side, which we may just as well have considered, is that the space consumption may be *underestimated* by considering, say, $\lceil \log S \rceil$ instead of S. Our answer to the former problem (namely, normalization—see Sect. 3) is essentially to stipulate that resources attain all natural-number values (hence, we permit S but not 2^S); the answer to the latter, the investigation of which we defer to future work, could be to consider the *number of ways* of attaining each natural-number value—for example, a value of $k \geq 1$ for $\lceil \log S \rceil$ follows from any one of $9 \times 10^{k-1}$ values for S, whereas we should like this number to be independent of k.

3 NORMALIZATION

To resolve inconsistencies such as that above, in which space can, depending on how it is measured (or even on how measured values are labelled), seem either more or less important than time, we introduce a notion of *normalization* of resource. We do this in such a way that S and T above are deemed normalized (and hence their mutual comparison is deemed valid), but S' is not (and hence the comparison between S' and T is deemed meaningless). After some thought, it seems natural—hopefully uniquely so—that S' should normalize to S; our definition satisfies this.

Note that S and S' differ in their respective *ranges*: assuming that there exist for Turing machine Φ from our example inputs x of all natural-number sizes, and supposing for simplicity that $S(x) = \sigma(x)$, we have that S maps surjectively to $\mathbb{N}$ and S' surjectively to $\{1, 2, 4, 8, \ldots\}$; the former property seems the more natural (quite literally!), and we use it as the model of our notion of normalization.

3.1 Definition of Normalization

Definition 4 ($\mathcal{C}$-normalized form; $\mathcal{C}$-normal). Let $\mathcal{C}$ be a class of computers.** For each $\Phi \in \mathcal{C}$, let X_Φ be the set of input values for Φ.

- Let A be a resource function that can take as its subscript any computing system Φ in $\mathcal{C}$ ($A_\Phi : X_\Phi \to \mathbb{N}$). Define the *$\mathcal{C}$-normalized form* of A to be the resource $A^{\mathcal{C}}$ given by $A^{\mathcal{C}}_\Phi : X_\Phi \to \mathbb{N}$,

$$A^{\mathcal{C}}_\Phi(x) := \left|\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_\Phi(x)\}\right| \quad . \quad (1)$$

** $\mathcal{C}$ may, for example, be the class of Turing machines, or a class of analogue computers (such as are introduced in [10] and [11]).

(In words, $A^{\mathcal{C}}_{\Phi}(x)$ is the number of distinct values less than $A_{\Phi}(x)$ taken by A, ranging over all computers in $\mathcal{C}$ and all input values. This is a measure of 'how much use A makes' of the natural numbers less than $A_{\Phi}(x)$.)

- Resource function A is *$\mathcal{C}$-normal* if $A = A^{\mathcal{C}}$ (by which we mean $A_{\Phi}(x) = A^{\mathcal{C}}_{\Phi}(x)$ for all $\Phi \in \mathcal{C}$ and $x \in X_{\Phi}$).

We now give an example, which relates to previous discussion, of this definition's use. Recall from our example (in Sect. 2.2: *The Problem: Dominance's Incompatibility with Unrestricted Resource*) the resources S and S', and let $\mathcal{T}$ be the class of Turing machines. We demonstrate that the $\mathcal{T}$-normalized form of S' (and of S) is S, i.e., that

$$S'^{\mathcal{T}} = S^{\mathcal{T}} = S \ .$$

Note that, for each $n \in \mathbb{N}$, there exists a Turing machine that, regardless of input, writes to exactly n distinct cells and then halts (for example, the machine may, by way of countdown, take n distinct states—once each—, in each state writing to a new cell; and then halt). Hence,

$$\{S_{\Psi}(y) \mid \Psi \in \mathcal{T} \wedge y \in X_{\Psi}\} = \mathbb{N} \ , \tag{2}$$

and so, by definition of S',

$$\{S'_{\Psi}(y) \mid \Psi \in \mathcal{T} \wedge y \in X_{\Psi}\} = \{2^n \mid n \in \mathbb{N}\} \ . \tag{3}$$

Hence, for any $\Phi \in \mathcal{T}$ and any $x \in X_{\Phi}$,

$$\begin{array}{rcl} S'^{\mathcal{T}}_{\Phi}(x) & \overset{(1)}{=} & |\{S'_{\Psi}(y) \mid \Psi \in \mathcal{T} \wedge y \in X_{\Psi} \wedge S'_{\Psi}(y) < S'_{\Phi}(x)\}| \\ & \overset{(3)}{=} & |\{2^n \mid n \in \mathbb{N} \wedge 2^n < S'_{\Phi}(x)\}| \\ & \overset{\text{defn. of } S'}{=} & \left|\left\{2^n \mid n \in \mathbb{N} \wedge 2^n < 2^{S_{\Phi}(x)}\right\}\right| \\ & = & \left|\left\{2^0, 2^1, \ldots, 2^{S_{\Phi}(x)-1}\right\}\right| \\ & = & S_{\Phi}(x) \end{array}$$

and

$$\begin{array}{rcl} S^{\mathcal{T}}_{\Phi}(x) & \overset{(1)}{=} & |\{S_{\Psi}(y) \mid \Psi \in \mathcal{T} \wedge y \in X_{\Psi} \wedge S_{\Psi}(y) < S_{\Phi}(x)\}| \\ & \overset{(2)}{=} & |\{n \in \mathbb{N} \mid n < S_{\Phi}(x)\}| \\ & = & |\{0, 1, \ldots, S_{\Phi}(x) - 1\}| \\ & = & S_{\Phi}(x) \ ; \end{array}$$

$S'^{\mathcal{T}} = S^{\mathcal{T}} = S$, as claimed.

Note that the inconsistency discussed in Sect. 2.2: *The Problem* ... would not have arisen had we stipulated that resources be $\mathcal{T}$-normal, in which case we may have considered resources S and T—of which T dominates—, but not S'. The criterion of $\mathcal{T}$-normality restricts our choice of resource such that the 'dominance engineering' of Sect. 2.2: *The Problem* ... is not possible.

3.2 Properties of Normalization

Theorem 1 (isotonicity). $\mathcal{C}$-normalization (by which we mean the taking of $\mathcal{C}$-normalized forms) is strictly isotone: if $A_{\Phi_1}(x_1) < A_{\Phi_2}(x_2)$, then $A^{\mathcal{C}}_{\Phi_1}(x_1) < A^{\mathcal{C}}_{\Phi_2}(x_2)$.

Proof. Suppose that $A_{\Phi_1}(x_1) < A_{\Phi_2}(x_2)$. For $i \in \{1,2\}$, let $I_i = \{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_{\Phi_i}(x_i)\}$ (so that $A^{\mathcal{C}}_{\Phi_i}(x_i) = |I_i|$). Since $A_{\Phi_1}(x_1) < A_{\Phi_2}(x_2)$, $I_1 \subseteq I_2$. Further, $A_{\Phi_1}(x_1) \in I_2 \setminus I_1$, so $I_1 \subsetneq I_2$. Hence, $A^{\mathcal{C}}_{\Phi_1}(x_1) = |I_1| < |I_2| = A^{\mathcal{C}}_{\Phi_2}(x_2)$, as required.[††] $\square$

Corollary 1. Consider the following statements.

(i)	$A_{\Phi_1}(x_1) < A_{\Phi_2}(x_2)$.	(i′)	$A^{\mathcal{C}}_{\Phi_1}(x_1) < A^{\mathcal{C}}_{\Phi_2}(x_2)$.
(ii)	$A_{\Phi_1}(x_1) = A_{\Phi_2}(x_2)$.	(ii′)	$A^{\mathcal{C}}_{\Phi_1}(x_1) = A^{\mathcal{C}}_{\Phi_2}(x_2)$.
(iii)	$A_{\Phi_1}(x_1) > A_{\Phi_2}(x_2)$.	(iii′)	$A^{\mathcal{C}}_{\Phi_1}(x_1) > A^{\mathcal{C}}_{\Phi_2}(x_2)$.

As a corollary to Theorem 1, we have that (i) $\Leftrightarrow$ (i′), (ii) $\Leftrightarrow$ (ii′) and (iii) $\Leftrightarrow$ (iii′).

Proof. Theorem 1 establishes (i) $\Rightarrow$ (i′) and, simply by exchanging the roles of subscripts '1' and '2', (iii) $\Rightarrow$ (iii′). (ii) $\Rightarrow$ (ii′) is clear: if (ii), then

$$\begin{aligned} A^{\mathcal{C}}_{\Phi_1}(x_1) &\overset{(1)}{=} |\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_{\Phi_1}(x_1)\}| \\ &\overset{(ii)}{=} |\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_{\Phi_2}(x_2)\}| \\ &\overset{(1)}{=} A^{\mathcal{C}}_{\Phi_2}(x_2) \ ; \end{aligned}$$

that is, if (ii), then (ii′).

From (i) $\Rightarrow$ (i′), (ii) $\Rightarrow$ (ii′) and (iii) $\Rightarrow$ (iii′), we obtain the respective contrapositives: $\neg$(i′) $\Rightarrow$ $\neg$(i), $\neg$(ii′) $\Rightarrow$ $\neg$(ii) and $\neg$(iii′) $\Rightarrow$ $\neg$(iii); and, by trichotomy of the natural numbers, we have that exactly one of (i) – (iii) and exactly one of (i′) – (iii′) hold. Hence, where (α), (β) and (γ) stand

[††] It should be noted that $|I_1| < |I_2|$ follows from $I_1 \subsetneq I_2$ *because* I_1 *and* I_2 *are finite*. This is the case since each I_i is a set of natural numbers *bounded above by* $A_{\Phi_i}(x_i) \in \mathbb{N}$.

for any permutation of (i), (ii) and (iii), and (α'), (β') and (γ') for the same permutation of (i′), (ii′) and (iii′), we have that

$$\begin{array}{lcl} (\alpha') & \overset{\text{trichotomy}}{\Rightarrow} & \neg(\beta') \wedge \neg(\gamma') \\ & \overset{\text{contrapositives}}{\Rightarrow} & \neg(\beta) \wedge \neg(\gamma) \\ & \overset{\text{trichotomy}}{\Rightarrow} & (\alpha) \quad . \end{array}$$

So (i′) ⇒ (i), (ii′) ⇒ (ii) and (iii′) ⇒ (iii). □

Theorem 2 (idempotence).

1. $\mathcal{C}$-normalization is idempotent: for all A and $\mathcal{C}$, $A^{\mathcal{C}\mathcal{C}} = A^{\mathcal{C}}$ (that is, $A^{\mathcal{C}\mathcal{C}}_{\Phi}(x) = A^{\mathcal{C}}_{\Phi}(x)$ for all $\Phi \in \mathcal{C}$ and $x \in X_{\Phi}$).

2. Hence, each $\mathcal{C}$-normalized form is $\mathcal{C}$-normal.

Proof.

1. Fix arbitrary elements Φ of $\mathcal{C}$ and x of X_{Φ}. Note that the sets

$$\{A_{\Psi}(y) \mid \Psi \in \mathcal{C} \wedge y \in X_{\Psi} \wedge A_{\Psi}(y) < A_{\Phi}(x)\}$$

and

$$\left\{A^{\mathcal{C}}_{\Psi}(y) \mid \Psi \in \mathcal{C} \wedge y \in X_{\Psi} \wedge A_{\Psi}(y) < A_{\Phi}(x)\right\}$$

have the same cardinality (this is by the Schröder-Bernstein theorem, which is evocable since Corollary 1 gives that the mappings $A_{\Psi}(y) \mapsto A^{\mathcal{C}}_{\Psi}(y)$ and $A^{\mathcal{C}}_{\Psi}(y) \mapsto A_{\Psi}(y)$ are injective); we use this equipollence below, at equality 'eq.'.

$$\begin{array}{lcl} A^{\mathcal{C}\mathcal{C}}_{\Phi}(x) & \overset{(1)}{=} & \left|\left\{A^{\mathcal{C}}_{\Psi}(y) \mid A^{\mathcal{C}}_{\Psi}(y) < A^{\mathcal{C}}_{\Phi}(x)\right\}\right| \\ & \overset{\text{Cor. 1}}{=} & \left|\left\{A^{\mathcal{C}}_{\Psi}(y) \mid A_{\Psi}(y) < A_{\Phi}(x)\right\}\right| \\ & \overset{\text{eq.}}{=} & \left|\{A_{\Psi}(y) \mid A_{\Psi}(y) < A_{\Phi}(x)\}\right| \\ & \overset{(1)}{=} & A^{\mathcal{C}}_{\Phi}(x) \quad . \end{array}$$

(For clarity, we suppress in this equation the fact that dummy variables Ψ and y are taken from $\mathcal{C}$ and X_{Ψ} respectively.) Hence, $A^{\mathcal{C}\mathcal{C}} = A^{\mathcal{C}}$, as required.

2. Consider an arbitrary $\mathcal{C}$-normalized form $A^{\mathcal{C}}$. By point 1, $A^{\mathcal{C}} = A^{\mathcal{C}\mathcal{C}}$, and so $A^{\mathcal{C}}$ is, by the second point of Definition 4, $\mathcal{C}$-normal. □

Notation 1 ($[[n]]$). For $n \in \mathbb{N}$, let $[[n]] = \{i \in \mathbb{N} \mid i < n\}$. Extend this notation by letting $[[\infty]] = \mathbb{N}$. Hence, $[[0]] = \emptyset$, $[[1]] = \{0\}$, $[[2]] = \{0, 1\}$, $\ldots$, $[[n]] = \{0, 1, \ldots, n-1\}$, $\ldots$, $[[\infty]] = \{0, 1, 2, \ldots\}$.

(The set-theoretically minded reader may wonder why we introduce $[[0]]$, $[[1]]$, $\ldots$, $[[\infty]]$ when there already exist ordinals $0, 1, \ldots, \omega$; we use the '$[[\cdot]]$' notation in order to retain the distinction between ordinals and cardinals.)

Lemma 1. Let X be any subset of $\mathbb{N}$. If, for any $i \in X$, $[[i]] \subseteq X$, then X is of the form $[[n]]$ for some $n \in \mathbb{N}^\infty := \mathbb{N} \cup \{\infty\}$.

Proof.

- If $X = \emptyset$, then the result is clear: $X = \emptyset = [[0]]$.
- If X is infinite, then for every $j \in \mathbb{N}$, there exists $i \in X$ such that $i > j$; then $j \in [[i]] \overset{\text{hypothesis}}{\subseteq} X$, and so $X = \mathbb{N} = [[\infty]]$.
- If X is finite and non-empty, then it has a maximal element, m, say. By hypothesis, $[[m]] \subseteq X$ (and so $[[m+1]] \subseteq X$ since $m \in X$), and, by maximality of m, $X \cap \{m+1, m+2, \ldots\} = \emptyset$. Hence, $X = [[m+1]]$.

These cases are exhaustive, and, in each, $X = [[n]]$ for some $n \in \mathbb{N}^\infty$. $\square$

Theorem 3 (normal $\Leftrightarrow$ onto $[[n]]$). A is $\mathcal{C}$-normal if and only if the image set (over all computers and input values) $\mathcal{A} := \{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi\}$ is $[[n]]$ for some $n \in \mathbb{N}^\infty$.

Proof. ('Only if' direction: 'A is $\mathcal{C}$-normal $\Rightarrow$ $\mathcal{A} = [[n]]$'.) Suppose that A is $\mathcal{C}$-normal. We wish to show that, for any $i \in \mathcal{A}$, $[[i]] \subseteq \mathcal{A}$, for then Lemma 1 gives that $\mathcal{A}$ is of the form $[[n]]$ for some $n \in \mathbb{N}^\infty$.

Suppose that $i \in \mathcal{A}$; say $A_\Phi(x) = i$, with $\Phi \in \mathcal{C}$ and $x \in X_\Phi$. Now,

$$\begin{array}{rcl} i & = & A_\Phi(x) \\ & \overset{\text{hypothesis}}{=} & A^{\mathcal{C}}_\Phi(x) \\ & \overset{(1)}{=} & |\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_\Phi(x)\}| \\ & = & |\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < i\}| \quad . \end{array}$$

So the set $\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < i\}$

- has cardinality i,
- is a set of natural numbers, and
- is strictly bounded above by i,

and, hence, this set is $[[i]]$; but, by definition of $\mathcal{A}$, this set can also be expressed as $\{j \in \mathcal{A} \mid j < i\}$, which is a subset of $\mathcal{A}$, and so $[[i]] \subseteq \mathcal{A}$, as required.

('If' direction: '$\mathcal{A} = [[n]] \Rightarrow A$ is $\mathcal{C}$-normal'.) Conversely, suppose that $\mathcal{A} = [[n]]$, where $n \in \mathbb{N}^\infty$. We wish to show that A is $\mathcal{C}$-normal.

Fix arbitrary $\Phi \in \mathcal{C}$ and $x \in X_\Phi$.

$$\begin{array}{rcl} A^{\mathcal{C}}_{\Phi}(x) & \overset{(1)}{=} & |\{A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_\Phi(x)\}| \\ & \overset{\text{defn. of } \mathcal{A}}{=} & |\{j \in \mathcal{A} \mid j < A_\Phi(x)\}| \\ & \overset{\text{hypothesis}}{=} & |\{j \in [[n]] \mid j < A_\Phi(x)\}| \\ & \overset{A_\Phi(x) \in \mathcal{A}}{=} & |[[A_\Phi(x)]]| \\ & = & A_\Phi(x) \quad . \end{array}$$

A is $\mathcal{C}$-normal, as required. □

3.3 Why Normalization?

As we have seen in Sect. 2.2: *The Problem* . . . , leaving unchecked our choice of resources diminishes the usefulness of dominance and related notions. This motivates our consideration of *$\mathcal{C}$-normal* resources, restriction to which restores those desirable properties for which such notions were introduced. This is further bolstered by the following intuition.

If we allow as a resource *any* function with codomain $\mathbb{N}$ (that satisfies Blum's axioms), then we are effectively dealing with 'cardinals': we are *counting* time steps, tape cells or similar; we have an intrinsic *unit of measurement*. This is resource-dependent and not conducive to comparison (e.g., how many time steps should we deem of equivalent cost to one tape cell?). If we allow only *$\mathcal{C}$-normal* resource functions, then we have 'ordinals': 0 represents the least resource consumption, 1 the second-least, 2 the third-least, and so on; this is independent of specific resources and of any units of measurement suggested thereby. Comparison (according to the '$\in \mathcal{O}$' pre-ordering) seems fair and equal-footed.

Normalization represents our proposed restriction of the notion of resource when dealing with non-Turing computers. Explicitly, we propose that an

analysis of the complexity of a non-standard computing device should feature normalization of any resources considered, before '$\in \mathcal{O}$' comparisons are made (e.g., with other computers' complexity functions).

4 DISCUSSION

When measuring the cost of performing a computation *by Turing machine*, our choice of resources is guided by decades of collective experience and intuition, and by Blum's axioms (of which the necessity is common sense, even if the sufficiency is arguable).

In the context of *non-Turing* computation, exactly what we should admit as a resource is a much less studied, much less understood problem. Starting with Blum's axioms seems reasonable, but we have seen that they alone are not enough. Accordingly, we apply further restriction; namely, that considered resources be $\mathcal{C}$*-normal*. Explicitly,

> in the context of complexity analyses of non-standard computers, we advocate the use only of resources
>
> - as described in Sect. 1.1;
> - that satisfy Blum's axioms[‡‡] (see Sect. 2.1); and
> - that are $\mathcal{C}$-normal (see Sect. 3), where $\mathcal{C}$ is the encompassing class of computers.

Such restriction precludes certain 'deceptive' complexity behaviour, such as 'dominance engineering' via application of quickly growing, isotone functions (whereby the importance of essentially insignificant resources may be exaggerated), and renders resource measures 'ordinal' rather than 'cardinal', allowing seemingly fairer comparison.

For these reasons, we take the above restrictions as the basis of our notion of resource in those projects—EP/G003017/1 and that described in [2]—of which the present work is part.

An obvious question for future study is

> *are Blum's axioms, together with $\mathcal{C}$-normality, enough,*

or are they still sufficiently permissive to admit resources with deceptive complexity behaviour? A superficial consideration suggests that undesirable

[‡‡] From here, recall, we get many standard results from [6]; resources as advocated here are, hence, a subset of those for which Blum's theorems hold.

behaviour *is* still accommodated: for example, transformations of resources may result in $\mathcal{C}$-normality *whilst not being order-preserving*. Such issues will be investigated in the context of the framework of [2] and EP/G003017/1. Another issue for future study is that Blum's axioms and $\mathcal{C}$-normality alone do not guarantee any desirable results concerning 'resource constructibility' (by which we have in mind the obvious generalization of time/space constructibility)—compare with the approach to constructibility of Kojiro Kobayashi.

5 ACKNOWLEDGEMENTS

We thank the International Journal of Unconventional Computing's reviewers for their extremely thorough and useful comments; Bob Coecke and Joël Ouaknine for their continued support and supervision; and participants of *Unconventional Computing 2007*, of the *Second International Workshop on Natural Computing*, and of *Quantum Physics and Logic/Development of Computational Models 2008* for their encouraging feedback and discussion. We acknowledge the generous support of the EPSRC; this work forms part of project EP/G003017/1.

REFERENCES

[1] A. Adamatzky, B. De Lacy Costello, L. Bull, S. Stepney, and C. Teuscher (editors). (2005 onwards). International Journal of Unconventional Computing. Old City Publishing.

[2] E. Blakey. (2008). A Model-Independent Theory of Computational Complexity; Price: From Patience to Precision (and Beyond).
Available at `http://users.ox.ac.uk/~quee1871/transfer.pdf`.

[3] E. Blakey. (2008). Computational Complexity in Non-Turing Models of Computation; the *What*, the *Why* and the *How*. Proceedings of *Quantum Physics and Logic/Development of Computational Models 2008* (to appear).
Available at `http://users.ox.ac.uk/~quee1871/dcm08.pdf`.

[4] E. Blakey. (2008). Dominance: Consistently Comparing Computational Complexity. Oxford University Computing Science Research Report CS-RR-08-09.

[5] E. Blakey. (2007). On the Computational Complexity of Physical Computing Systems. Proceedings of *Unconventional Computing 2007* pp. 95–115.

[6] M. Blum. (1967). A Machine-Independent Theory of the Complexity of Recursive Functions. J. of the Assoc. for Computing Machinery **14** no. 2 pp. 322–336.

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

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

[9] C. Papadimitriou. (1995). Computational Complexity. Addison-Wesley.

[10] M. B. Pour-El. (1974). Abstract Computability and Its Relation to the General Purpose Analog Computer (Some Connections Between Logic, Differential Equations and Analog Computers). Trans. Am. Math. Soc. **199** pp. 1–28.

[11] C. E. Shannon. (1941). Mathematical Theory of the Differential Analyzer. J. Math. Phys. MIT **20** pp. 337–354.