INTERNATIONAL JOURNAL OF COMPUTERS
Issue 4, Volume 4, 2010

Apples & Oranges? Comparing Unconventional Computers

Ed Blakey

***Abstract*—** Complexity theorists routinely compare—via the pre-ordering induced by asymptotic notation—the efficiency of computers so as to ascertain which offers the most efficient solution to a given problem. Tacit in this statement, however, is that the computers conform to a *standard computational model*: that is, they are Turing machines, random-access machines or similar. However, whereas meaningful comparison between these *conventional* computers is well understood and correctly practised, that of *non-standard* machines (such as quantum, chemical and optical computers) is rarely even attempted and, where it is, is often attempted under the typically false assumption that the conventional-computing approach to comparison is adequate in the unconventional-computing case. We discuss in the present paper a computational-model-independent approach to the comparison of computers' complexity (and define the corresponding complexity classes). Notably, the approach allows meaningful comparison between an unconventional computer and an existing, digital-computer benchmark that solves the same problem.

***Keywords*—** Computational complexity, computational resource, dominance, theoretical computer science, unconventional computation.

I. Introduction

COMPUTATIONAL complexity theory has as one of its chief aims the quantification of mathematical problems' difficulty: complexity theorists wish to make statements of the form

> 'solving problem X (e.g., the Travelling Salesperson Problem, factorization or addition) requires $\mathcal{O}(f)$ time, $\mathcal{O}(g)$ space, etc.'

(The 'etc.' here refers to computational resources other than run-time and memory space—see [1], [4] and Sect. II-A.) However, it is possible directly to measure the complexity not of *problems* but only of *methods that solve these problems*: whereas one would like to demonstrate that

> 'problem X requires $\mathcal{O}(f)$ time',

it is usually forthcoming only that

> 'problem X can be solved *by algorithm Y*, which requires $\mathcal{O}(f)$ time'.

Typically, then, all that is known about a *problem's* complexity is that it is bounded above by that of the most efficient, known *solution method* for the problem.

(As an aside, we recall the novel approach given in [7] to problems similar to the Travelling Salesperson Problem.)

It is clear from this that the ability to *compare* computers' efficiency—and thereby to ascertain which computer offers the most efficient solution to a problem—is of the utmost importance. It is unsurprising, then, that, *when the computers in question are 'standard'* (e.g., when they are modelled as Turing machines), the method by which they may be compared is well understood; we now recap this method, by first recalling the auxiliary notions of $\mathcal{O}$-notation and the pre-ordering induced thereby.

Definition 1:
- Let $\mathcal{O}(g(n))$ denote the class of all functions $f(n)$ such that there exist a threshold $n_0 \in \mathbb{N}$ and constant $c \in \mathbb{R}$ satisfying $|f(n)| \leq c\,|g(n)|$ for all natural numbers n such that $n > n_0$.
- Write '$f \lesssim g$' for '$f \in \mathcal{O}(g)$', and
- '$f \not\lesssim g$' for '$f \notin \mathcal{O}(g)$'.

It is trivial that $\lesssim$ is both *reflexive* (since threshold 0 and constant 1 witness that $f \lesssim f$) and *transitive* (since, if threshold-constant pair (n_0, c) witnesses that $f \lesssim g$ and (n_0', c') that $g \lesssim h$, then $(\max\{n_0, n_0'\}, cc')$ witnesses that $f \lesssim h$); hence, $\lesssim$ is a *pre-ordering* of functions. Of specific interest here is that $\lesssim$ may be used as a pre-ordering of *complexity* functions (which we define in Definition 3). We can now describe the method by which the efficiency of standard computers is compared.

Suppose that (conventional, Turing-machine-style) computers Φ and Ψ have respective time-complexity functions T_Φ^* and T_Ψ^* (that is, given an arbitrary input value of size n, Φ requires at most $T_\Phi^*(n)$ time steps to return the corresponding output value; similarly Ψ—see Definition 3). One may utilize the pre-ordering $\lesssim$ to determine which (if either) of Φ and Ψ is the more efficient, in the obvious way:
- if $T_\Phi^* \lesssim T_\Psi^* \not\lesssim T_\Phi^*$, then Φ is deemed the more efficient computer;
- if $T_\Phi^* \not\lesssim T_\Psi^* \lesssim T_\Phi^*$, then Ψ is deemed the more efficient computer;
- if $T_\Phi^* \lesssim T_\Psi^* \lesssim T_\Phi^*$, then Φ and Ψ are deemed equally efficient; and
- if $T_\Phi^* \not\lesssim T_\Psi^* \not\lesssim T_\Phi^*$, then Φ and Ψ are deemed incomparably efficient.

(These computers, one supposes, solve the same problem, for else, if they solved different problems, then little meaning could be attributed to a comparison of their efficiency. One may, prima facie, assume that such comparison *would* be meaningful, that it would say something, for example, about which problem is harder to solve, but a problem's complexity is defined in terms of an *optimal*—not *arbitrary*—solution method's complexity; comparison of *problems* is

We acknowledge the generous financial support of the Engineering and Physical Sciences Research Council: this work is funded by the grant *Complexity and Decidability in Unconventional Computational Models* (EP/G003017/1).

better achieved using the standard, complexity-theoretic notion of *reduction*.)

Note that we tacitly identify (overall) efficiency with *time* efficiency in particular. This is because, in the case of conventional, Turing-style computers, the only computational resources that need be considered are *time* and *space* (recall also the approach of [10]), and, of these, time is always consumed in greater quantity (the space complexity S^*_Φ of a conventional computer Φ is always a smaller function than the time complexity T^*_Φ—in that $S^*_\Phi(n) \leq T^*_\Phi(n)$ for all n—, since writing to a tape cell takes one time step).

However, there exist *unconventional* computers (conforming, for example, to quantum—and in addition to the quantum model as a whole, one may consider the quantum generalization of the Turing machine; see, for example, [6]—, chemical, analogue or genetic-algorithm—recall in relation to this latter example [9]—paradigms), for which this is not the case: such computers may consume resources other than time and space, whence it is no longer valid to identify time efficiency with overall efficiency; consequently, one cannot simply apply the pre-ordering $\lesssim$ to these unconventional computers' time-complexity functions in order to determine which is the most efficient. For example, we describe in [2] an analogue computer that can factorize natural numbers in time and space *polynomial* in the size of the input value; since the true (exponential) complexity of the system lies in its *precision complexity* (which notion is introduced in [3] and discussed in [2]), an exclusively time-based consideration—à la standard, Turing-machine complexity theory—of the system leads to an overly generous quantification of its complexity and to misleading comparisons with other (e.g., Turing-machine) solutions to the problem of factorization.

In summary, then, the difficulty is that unconventional computers may consume unconventional resources (in addition to the standard resources of time and space), which leads to each computer's having many complexity functions. The comparison of the efficiency of two such computers, then, is not merely a case of applying the pre-ordering $\lesssim$ to the respective time-complexity functions. We describe in this paper a more suitable method whereby such computers' efficiency can be meaningfully compared.

II. Comparison of Computers

A. Resource

We mention above the widely considered computational resources of *time* and *space*. We also hint that certain (unconventional) computers consume other resources (such as precision). We defer to [1] and [4] a full motivation and discussion of resources, making instead the following definition, which is adequate for present purposes.

Definition 2:

- We model a *resource* as a function (denoted by an upper-case letter A, B, C, etc.) that depends upon the choice of *computational system* (shown as a subscript to the function) and that maps each *input value* to a natural number, which can be thought of as the corresponding number of units of the resource consumed by the system in processing the input value.
- Let T denote the resource of *time*,
- S *space*, and
- P *precision*.

Hence, for computer Φ and input value x, $A_\Phi(x)$ denotes the amount of resource A consumed by Φ in processing x. Specific and notable examples are $T_\Phi(x)$, which denotes the number of units of *time* (e.g., the number of time steps if Φ is a Turing machine) taken by Φ to process x, and $S_\Phi(x)$, the number of units of *space* occupied (e.g., the number of tape cells used if Φ is a Turing machine); the resource of *precision* is discussed in, for example, [3].

B. Complexity

Given a computer, and considering a specific resource, we may ask how this resource scales. In particular, we may be interested not in the resource used/required by the computer given *one specific input value* (that is, in some '$A_\Phi(x)$'), but in the resource used *as a function of the size of the input value*—this is what is meant by the *complexity function* corresponding to the resource. Specifically, we have the following.

Definition 3: For resource A, the corresponding *complexity function* A^* is defined by

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

In this definition, $|x|$ stands for the *size* of input value x. If, for example, x is a natural number 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 $\log_2(x)$ to this number of bits).

Note that our usage in Sect. I of 'T^*' and 'S^*' conforms with the notation of Definition 3.

C. Dominance Motivated

We see in Sect. I that $\mathcal{O}$-notation, and in particular the pre-ordering $\lesssim$ induced thereby, allows comparison of the respective time-complexity functions of standard (e.g., Turing-machine) computers (in this standard context, the comparison of *time*-efficiency actually offers an assessment of the relative, *overall* efficiency of the computers being analysed, as we comment in Sect. I). More generally, the pre-ordering allows 'apples-to-apples' comparison, in that computers Φ and Ψ may be compared *with respect to the* ***same*** (arbitrarily chosen) *resource* A.

However, when one considers unconventional (optical, quantum, DNA, etc.) computers, which may consume many unconventional resources (precision, energy, thermodynamic cost, etc.), it is no longer generally true that an 'apples-to-apples' comparison with respect to any given resource equates to a fair comparison of the computers' *overall* efficiency. We now illustrate this phenomenon by recalling an instance from [3].

Supposing that we wish to find the *greatest common divisor* of two given, natural numbers (with a combined length of n

digits, say), we have available (amongst others) two solution methods:

- *Euclid's Algorithm* (which we denote by 'E'), which has time and space complexities polynomial, and precision complexity constant, in n (that is, $T^*_{\mathrm{E}} \in \mathcal{O}\left(n^k\right)$ (Lemma 11.7 of [8]), $S^*_{\mathrm{E}} \in \mathcal{O}\left(n^k\right)$ (since $S^*_{\mathrm{E}}(n) \leq T^*_{\mathrm{E}}(n)$ for all n) and $P^*_{\mathrm{E}} \in \mathcal{O}(1)$ (Theorem 1 of [3]) (k constant)); and
- an *analogue system* (which we denote by 'B', and of which we defer description to [3]), which has time and space complexities constant, and precision complexity exponential, in n (that is, $T^*_{\mathrm{B}}, S^*_{\mathrm{B}} \in \mathcal{O}(1)$ and $P^*_{\mathrm{B}} \in \mathcal{O}(l^n)$ (see [3]) (l constant)).

(As an aside, see [5] for a cryptographic application of the algorithm E.)

One may describe the methods E and B respectively as polynomial- and constant-time (and infer by 'apples-to-apples' comparison that B is the more time-efficient); however, it is intuitively more insightful to describe the latter as an *exponential-precision* (and, hence, less efficient overall), rather than *constant-time*, method, since the former description focuses on the more 'relevant' resource. We now define *dominance* so as to formalize this notion of 'relevance', and to allow meaningful comparison between computers regardless of how many different resources they consume.

D. Dominance Defined

The intent of dominance is that the complexity functions corresponding to resources deemed to be dominant should be $\lesssim$-greater than those corresponding to other resources; non-dominant resources, then, are negligible in the sense that their asymptotic contribution to a computer's overall resource consumption is dwarfed by that of dominant resources. Accordingly, we make the following tentative definition.

Definition 4 (provisional—see Definition 5): A *dominant resource* for a computer Φ is a resource A such that, for any resource B, $B^*_\Phi \lesssim A^*_\Phi$.

This definition requires modification for the following two reasons.

- First, A's dominance is over *every* other resource B. It is not sufficient (in order that, for example, precision is shown to be dominant according to Definition 4) to show that precision complexity $\lesssim$-exceeds time and space complexity; rather, precision complexity must be shown to $\lesssim$-exceed time, space, and *all other conceivable resources'* complexity, of which resources there are indeterminately many—this is clearly a futile task and a worthless definition. Accordingly, we redefine dominance below *relative to an explicit set of resources*.
- Secondly, Definition 4 does not for every computer guarantee the existence of a dominant resource (much as we should like one), since there exist pairs of functions f and g such that $f \not\lesssim g \not\lesssim f$. We weaken accordingly the definition of dominance (essentially from 'A^* $\lesssim$-exceeds all other complexity functions' to 'A^* $\lesssim$-exceeds all other complexity functions *with which it is $\lesssim$-comparable*').

Definition 5 (to replace Definition 4): Let Φ be a computer, and let $\mathcal{R}$ be a finite, non-empty set of resources consumed by Φ (this consumption may be null: important is that it is *meaningful to ask* how much of a resource is consumed, even though the answer may be 'none'). An *$\mathcal{R}$-dominant resource* for Φ is a resource $A \in \mathcal{R}$ such that, for any resource $B \in \mathcal{R}$ satisfying $A^*_\Phi \lesssim B^*_\Phi$, we have that $B^*_\Phi \lesssim A^*_\Phi$.

We note in passing that $\mathcal{R}$-dominance has the following 'all-or-nothing' property.

Proposition 6: Let Φ and $\mathcal{R}$ be as in Definition 5, and let $X, Y \in \mathcal{R}$. Suppose that $X^*_\Phi \lesssim Y^*_\Phi \lesssim X^*_\Phi$. Then X is $\mathcal{R}$-dominant for Φ if and only if Y is.

Proof: Suppose that $X, Y \in \mathcal{R}$ are such that $X^*_\Phi \lesssim Y^*_\Phi \lesssim X^*_\Phi$.

(We prove first the 'X dominant $\Rightarrow$ Y dominant' direction.) Suppose that X is $\mathcal{R}$-dominant. Then, by Definition 5, $X^*_\Phi \lesssim B^*_\Phi \Rightarrow B^*_\Phi \lesssim X^*_\Phi$ for all $B \in \mathcal{R}$. If, for some $B \in \mathcal{R}$, $Y^*_\Phi \lesssim B^*_\Phi$, then $X^*_\Phi \lesssim B^*_\Phi$ (by the fact that $X^*_\Phi \lesssim Y^*_\Phi$ and by transitivity of '$\lesssim$'), whence $B^*_\Phi \lesssim X^*_\Phi$ (by $\mathcal{R}$-dominance of X), whence $B^*_\Phi \lesssim Y^*_\Phi$ (again by the fact that $X^*_\Phi \lesssim Y^*_\Phi$ and by transitivity of '$\lesssim$'). Since this holds for arbitrary $B \in \mathcal{R}$, we have that Y is $\mathcal{R}$-dominant.

('Y dominant $\Rightarrow$ X dominant' direction.) The converse argument differs only in that the roles of X and Y are switched, which is valid since the hypotheses of the proposition are symmetrical in X and Y. ■

It is at this juncture natural to introduce the following complexity classes.

Definition 7: Let $\mathcal{R}$ be as in Definition 5.

- For $A \in \mathcal{R}$ and function f, let $\mathsf{C}_{\mathcal{R}}(f, A)$ denote the complexity class of problems of which each is solved by some deterministic computer Φ with $\mathcal{R}$-dominant resource A such that $A^*_\Phi \lesssim f$.
- Let $\mathsf{NC}_{\mathcal{R}}(f, A)$ denote the analogous *non-deterministic* class.
- Let $\mathsf{C}_{\mathcal{R}}(f) = \bigcup_{R \in \mathcal{R}} \mathsf{C}_{\mathcal{R}}(f, R)$.
- Let $\mathsf{NC}_{\mathcal{R}}(f) = \bigcup_{R \in \mathcal{R}} \mathsf{NC}_{\mathcal{R}}(f, R)$.

Note, then, that we model non-determinism as a feature of computational paradigms, rather than as a resource function—see [4] for further discussion of this issue.

We define also a notion of 'overall complexity'.

Definition 8: Let Φ and $\mathcal{R}$ be as in Definition 5.

- Define $\mathcal{B}_{\mathcal{R},\Phi}$ by

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

Call this the *$\mathcal{R}$-complexity* of Φ; it is intended to capture, in a single complexity function, the 'overall complexity' of Φ.

- Let $\mathsf{B}_{\mathcal{R}}(f)$ denote the complexity class of problems for each of which there exists a deterministic computer Φ with $\mathcal{B}_{\mathcal{R},\Phi}(n) \leq f(n)$ for all n.
- Let $\mathsf{NB}_{\mathcal{R}}(f)$ denote the analogous *non-deterministic* class.

III. Complexity Class Inclusions

So as to give a flavour of the structure of the hierarchy of complexity classes (namely, $\mathsf{C}_{\mathcal{R}}(f,A)$, $\mathsf{C}_{\mathcal{R}}(f)$, $\mathsf{B}_{\mathcal{R}}(f)$, and their non-deterministic counterparts) defined in Definitions 7 and 8, we state and prove the following theorems.

We note first (in Theorem 9) the connection between *determinism* and *non-determinism* (specifically, that deterministic computers are a special case of non-deterministic computers).

Theorem 9: If $\mathcal{R}$ is a set of resources with $A \in \mathcal{R}$, and if f is a function, then

- $\mathsf{C}_{\mathcal{R}}(f,A) \subseteq \mathsf{NC}_{\mathcal{R}}(f,A)$,
- $\mathsf{C}_{\mathcal{R}}(f) \subseteq \mathsf{NC}_{\mathcal{R}}(f)$ and
- $\mathsf{B}_{\mathcal{R}}(f) \subseteq \mathsf{NB}_{\mathcal{R}}(f)$.

Proof: This follows directly from the fact that each deterministic computer is automatically a non-deterministic computer. ■

We consider now the dependency of our classes $\mathsf{C}_{\mathcal{R}}(f,A)$, etc. upon their *resource-set parameter* $\mathcal{R}$,

- first (in Theorem 10, Corollary 11 and Theorem 12) by contrasting a *subset* $\mathcal{R}$ against a *superset* $\mathcal{S}$,
- secondly (in Theorem 13) by contrasting *disjoint* sets $\mathcal{R}$ and $\mathcal{S}$, and
- thirdly (in Theorem 14) by contrasting *non-disjoint*, *non-enveloping* sets $\mathcal{R}$ and $\mathcal{S}$.

Theorem 10: If $\mathcal{S}$ is a resource set and $A \in \mathcal{R} \subseteq \mathcal{S}$, then

- $\mathsf{C}_{\mathcal{S}}(f,A) \subseteq \mathsf{C}_{\mathcal{R}}(f,A)$ and
- $\mathsf{NC}_{\mathcal{S}}(f,A) \subseteq \mathsf{NC}_{\mathcal{R}}(f,A)$.

Proof: Suppose that problem π is in class $\mathsf{C}_{\mathcal{S}}(f,A)$. Then, by definition, π is solved by a deterministic computer Φ for which resource A is $\mathcal{S}$-dominant and such that $A^* \lesssim f$. Since $A \in \mathcal{R} \subseteq \mathcal{S}$, A is also *$\mathcal{R}$-dominant*: that A is $\mathcal{S}$-dominant gives that no $B \in \mathcal{S}$ has the property that $A^* \lesssim B^* \not\lesssim A^*$, so certainly no $B \in \mathcal{R} \subseteq \mathcal{S}$ has this property, whence A is $\mathcal{R}$-dominant for Φ. Hence, $\pi \in \mathsf{C}_{\mathcal{R}}(f,A)$, as required.

Similarly, if problem ρ is in class $\mathsf{NC}_{\mathcal{S}}(f,A)$, then, by definition, ρ is solved by a non-deterministic computer Ψ for which resource A is $\mathcal{S}$-dominant and such that $A^* \lesssim f$. Since $A \in \mathcal{R} \subseteq \mathcal{S}$, A is also *$\mathcal{R}$-dominant*: that A is $\mathcal{S}$-dominant gives that no $B \in \mathcal{S}$ has the property that $A^* \lesssim B^* \not\lesssim A^*$, so certainly no $B \in \mathcal{R} \subseteq \mathcal{S}$ has this property, whence A is $\mathcal{R}$-dominant for Ψ. Hence, $\rho \in \mathsf{NC}_{\mathcal{R}}(f,A)$, as required. ■

As a consequence, we have the following.

Corollary 11: If $\mathcal{S}$ is a resource set and $\mathcal{R} \subseteq \mathcal{S}$, then

- $\mathsf{C}_{\mathcal{S}}(f) \subseteq \mathsf{C}_{\mathcal{R}}(f)$ and
- $\mathsf{NC}_{\mathcal{S}}(f) \subseteq \mathsf{NC}_{\mathcal{R}}(f)$.

Proof: If $\pi \in \mathsf{C}_{\mathcal{S}}(f) = \bigcup_{R\in\mathcal{S}} \mathsf{C}_{\mathcal{S}}(f,R)$, then $\pi \in \mathsf{C}_{\mathcal{S}}(f,A)$ for some $A \in \mathcal{S}$. Hence, by Theorem 10, $\pi \in \mathsf{C}_{\mathcal{R}}(f,A)$, and so $\pi \in \bigcup_{R\in\mathcal{R}} \mathsf{C}_{\mathcal{R}}(f,R) = \mathsf{C}_{\mathcal{R}}(f)$.

Similarly, if $\rho \in \mathsf{NC}_{\mathcal{S}}(f) = \bigcup_{R\in\mathcal{S}} \mathsf{NC}_{\mathcal{S}}(f,R)$, then $\rho \in \mathsf{NC}_{\mathcal{S}}(f,A)$ for some $A \in \mathcal{S}$. Hence, by Theorem 10, $\rho \in \mathsf{NC}_{\mathcal{R}}(f,A)$, and so $\rho \in \bigcup_{R\in\mathcal{R}} \mathsf{NC}_{\mathcal{R}}(f,R) = \mathsf{NC}_{\mathcal{R}}(f)$. ■

Theorem 12: If $\mathcal{S}$ is a resource set, and if $\mathcal{R} \subseteq \mathcal{S}$, then $\mathcal{B}_{\mathcal{R},\Phi} \lesssim \mathcal{B}_{\mathcal{S},\Phi}$ for all computers Φ. It does not necessarily hold, however, that $\mathsf{B}_{\mathcal{S}}(f) \subseteq \mathsf{B}_{\mathcal{R}}(f)$ for bounding function f, and neither do we have the converse class inclusion; similarly, it necessarily holds neither that $\mathsf{NB}_{\mathcal{S}}(f) \subseteq \mathsf{NB}_{\mathcal{R}}(f)$ nor conversely.

Proof: We consider a fixed computer, the corresponding subscripts of which we omit. Let $\bar{\mathcal{R}} = \{ A \in \mathcal{R} \mid A \text{ is } \mathcal{R}\text{-dominant} \}$, and define $\bar{\mathcal{S}}$ analogously; suppose that $\bar{\mathcal{R}} = \left\{A_1, \ldots, A_{|\bar{\mathcal{R}}|}\right\}$. Now, for each $A_i \in \bar{\mathcal{R}}$, there exists $B_i \in \bar{\mathcal{S}}$ such that $A_i^* \lesssim B_i^*$ (if $A_i \in \bar{\mathcal{S}}$, then this B_i may be taken to be A_i itself; else, if $A_i \notin \bar{\mathcal{S}}$, then A_i is, by definition of $\bar{\mathcal{S}}$, $\lesssim$-smaller than some element—which we may take as B_i—of $\bar{\mathcal{S}}$); hence, there exists a mapping $d: \bar{\mathcal{R}} \to \bar{\mathcal{S}}: A_i \mapsto B_i$ (where non-uniqueness of B_i given A_i is resolved via arbitrary choice, which does not require the axiom of choice since $\bar{\mathcal{S}}$ is finite) such that $A_i^* \lesssim d(A_i)^*$. Now, for each $B \in \bar{\mathcal{S}}$, B^* is an addend of $\mathcal{B}_{\mathcal{S}}$ (by definition, $\mathcal{B}_{\mathcal{S}} = \sum_{B\in\bar{\mathcal{S}}} B^*$), and so $B^* \leq \mathcal{B}_{\mathcal{S}}$ (by non-negativity of the other addends); in particular, $d(A_i)^* \leq \mathcal{B}_{\mathcal{S}}$ for all $A_i \in \bar{\mathcal{R}}$. Hence, $\sum_{i=1}^{|\bar{\mathcal{R}}|} d(A_i)^* \leq |\bar{\mathcal{R}}| \mathcal{B}_{\mathcal{S}}$, whence $\mathcal{O}\left(\sum_{i=1}^{|\bar{\mathcal{R}}|} d(A_i)^*\right) \subseteq \mathcal{O}\left(|\bar{\mathcal{R}}|\mathcal{B}_{\mathcal{S}}\right) = \mathcal{O}(\mathcal{B}_{\mathcal{S}})$; so $\mathcal{B}_{\mathcal{R}} = \sum_{i=1}^{|\bar{\mathcal{R}}|} A_i^* \in \mathcal{O}\left(\sum_{i=1}^{|\bar{\mathcal{R}}|} d(A_i)^*\right) \subseteq \mathcal{O}(\mathcal{B}_{\mathcal{S}})$. $\mathcal{B}_{\mathcal{R}} \in \mathcal{O}(\mathcal{B}_{\mathcal{S}})$, as required.

Now, despite this relation between $\mathcal{B}_{\mathcal{R}}$ and $\mathcal{B}_{\mathcal{S}}$, it may not be the case that $\mathsf{B}_{\mathcal{S}}(f) \subseteq \mathsf{B}_{\mathcal{R}}(f)$ (or that $\mathsf{B}_{\mathcal{R}}(f) \subseteq \mathsf{B}_{\mathcal{S}}(f)$). Consider a deterministic computer and resource sets $\mathcal{R}$ and $\mathcal{S}$ such that $\mathcal{B}_{\mathcal{R}}: n \mapsto 10$ and $\mathcal{B}_{\mathcal{S}}: n \mapsto n^2$. Note that $\mathcal{B}_{\mathcal{R}} \lesssim \mathcal{B}_{\mathcal{S}}$, but that $\mathcal{B}_{\mathcal{R}} \not\leq \mathcal{B}_{\mathcal{S}}$ (specifically when $n < 4$). Let $f: n \mapsto 20$ and $f': n \mapsto 2n^2$. Then we have that $\mathcal{B}_{\mathcal{R}} \leq f$, $\mathcal{B}_{\mathcal{R}} \not\leq f'$, $\mathcal{B}_{\mathcal{S}} \leq f'$, and $\mathcal{B}_{\mathcal{S}} \not\leq f$; so $\mathsf{B}_{\mathcal{S}}(f') \not\subseteq \mathsf{B}_{\mathcal{R}}(f')$ and $\mathsf{B}_{\mathcal{R}}(f) \not\subseteq \mathsf{B}_{\mathcal{S}}(f)$ (in each case, our deterministic computer places the corresponding problem in the former but not the latter class).

By considering instead a *non-deterministic* computer, we obtain the analogous results for NB rather than B. ■

Theorem 13: Let $\mathcal{R}$ and $\mathcal{S}$ be disjoint resource sets. Then no relation of class inclusion (neither '$\subseteq$' nor '$\supseteq$') holds in generality between

- $\mathsf{C}_{\mathcal{R}}(f,A)$ and $\mathsf{C}_{\mathcal{S}}(f,A)$;

similarly between

- $\mathsf{NC}_{\mathcal{R}}(f,A)$ and $\mathsf{NC}_{\mathcal{S}}(f,A)$,
- $\mathsf{C}_{\mathcal{R}}(f)$ and $\mathsf{C}_{\mathcal{S}}(f)$,
- $\mathsf{NC}_{\mathcal{R}}(f)$ and $\mathsf{NC}_{\mathcal{S}}(f)$,
- $\mathsf{B}_{\mathcal{R}}(f)$ and $\mathsf{B}_{\mathcal{S}}(f)$, and
- $\mathsf{NB}_{\mathcal{R}}(f)$ and $\mathsf{NB}_{\mathcal{S}}(f)$.

Neither is it generally the case that $\mathcal{B}_{\mathcal{R}} \lesssim \mathcal{B}_{\mathcal{S}}$ or vice versa, nor that $\mathcal{B}_{\mathcal{R}} \leq \mathcal{B}_{\mathcal{S}}$ or vice versa.

Here and in subsequent theorems, '$\leq$' as a relation between functions is to be interpreted *pointwise*: '$f \leq g$' stands for '$f(x) \leq g(x)$ for all x'.

Proof: Suppose, for a contradiction, that such an inclusion were to hold for all disjoint pairs $(\mathcal{R},\mathcal{S})$ of resource sets. So either $\mathsf{X}_{\mathcal{R}} \subseteq \mathsf{X}_{\mathcal{S}}$ or $\mathsf{X}_{\mathcal{R}} \supseteq \mathsf{X}_{\mathcal{S}}$ (where $\mathsf{X}_{\mathcal{U}}$ stands for some element of the set $\mathcal{X}_{\mathcal{U}} := \{\mathsf{C}_{\mathcal{U}}(f,A), \mathsf{NC}_{\mathcal{U}}(f,A), \mathsf{C}_{\mathcal{U}}(f), \mathsf{NC}_{\mathcal{U}}(f), \mathsf{B}_{\mathcal{U}}(f), \mathsf{NB}_{\mathcal{U}}(f)\}$ of complexity classes); and, whichever is the case, we see by reversing the roles of $\mathcal{R}$ and $\mathcal{S}$ (valid since, clearly, $\mathcal{R}$ and $\mathcal{S}$ are disjoint if and only if $\mathcal{S}$ and $\mathcal{R}$ are) that the other must also hold. Hence, $\mathsf{X}_{\mathcal{R}} = \mathsf{X}_{\mathcal{S}}$ for all disjoint $\mathcal{R}$ and $\mathcal{S}$.

Consider a computer that has constant R-complexity and exponential S-complexity (for resources R and S). Let $\mathcal{R} = \{R\}$ and $\mathcal{S} = \{S\}$; these are clearly disjoint. Then the equality $\mathsf{X}_{\mathcal{R}} = \mathsf{X}_{\mathcal{S}}$ cannot hold when $\mathsf{X}_{\mathcal{R}}$ stands for any of the classes belonging to $\mathcal{X}_{\mathcal{U}}$, as a consideration of resource-constructible functions shows. This is the sought contradiction.

Resource-constructibility aside, the claims concerning $\mathcal{B}_{\mathcal{R}}$ are clear from consideration of the computer of the previous paragraph. ∎

Theorem 14: Let $\mathcal{R}$ and $\mathcal{S}$ be such that $\mathcal{R} \nsubseteq \mathcal{S} \nsubseteq \mathcal{R}$ and $\mathcal{R} \cap \mathcal{S} \neq \emptyset$. Then no relation of class inclusion (neither '$\subseteq$' nor '$\supseteq$') holds in generality between

- $\mathsf{C}_{\mathcal{R}}(f,A)$ and $\mathsf{C}_{\mathcal{S}}(f,A)$;

similarly between

- $\mathsf{NC}_{\mathcal{R}}(f,A)$ and $\mathsf{NC}_{\mathcal{S}}(f,A)$,
- $\mathsf{C}_{\mathcal{R}}(f)$ and $\mathsf{C}_{\mathcal{S}}(f)$,
- $\mathsf{NC}_{\mathcal{R}}(f)$ and $\mathsf{NC}_{\mathcal{S}}(f)$,
- $\mathsf{B}_{\mathcal{R}}(f)$ and $\mathsf{B}_{\mathcal{S}}(f)$, and
- $\mathsf{NB}_{\mathcal{R}}(f)$ and $\mathsf{NB}_{\mathcal{S}}(f)$.

Neither is it generally the case that $\mathcal{B}_{\mathcal{R}} \lesssim \mathcal{B}_{\mathcal{S}}$ or vice versa, nor that $\mathcal{B}_{\mathcal{R}} \leq \mathcal{B}_{\mathcal{S}}$ or vice versa.

Proof: Given any pair of disjoint, non-empty resource sets $\mathcal{R}'$ and $\mathcal{S}'$, we may define $\mathcal{R} = \mathcal{R}' \cup \{\mathbf{0}\}$ and $\mathcal{S} = \mathcal{S}' \cup \{\mathbf{0}\}$ where $\mathbf{0}$ is the *null resource* that is identically equal to zero (for all computers and input values). Since the classes under consideration ($\mathsf{C}_{\mathcal{R}}(f,A)$, $\mathsf{NC}_{\mathcal{R}}(f,A)$, $\mathsf{C}_{\mathcal{R}}(f)$, $\mathsf{NC}_{\mathcal{R}}(f)$, $\mathsf{B}_{\mathcal{R}}(f)$ and $\mathsf{NB}_{\mathcal{R}}(f)$) are defined in terms of *dominant* resources, and since all resources dominate $\mathbf{0}$ (whence $\mathcal{R}$ and $\mathcal{R}'$ have the same dominant resources, as do $\mathcal{S}$ and $\mathcal{S}'$), we have that the results of the previous case (in which $\mathcal{R} \cap \mathcal{S} = \emptyset$) hold also for $\mathcal{R}$ and $\mathcal{S}$ as just constructed, which satisfy $\mathcal{R} \nsubseteq \mathcal{S} \nsubseteq \mathcal{R}$ (assuming that neither $\mathcal{R}'$ nor $\mathcal{S}'$ is equal to $\{\mathbf{0}\}$, which assumption is not problematic) and $\mathcal{R} \cap \mathcal{S} \neq \emptyset$. ∎

We consider now the dependency of our classes $\mathsf{C}_{\mathcal{R}}(f,A)$, etc. upon their *bounding-function parameter* f,

- first (in Theorem 15, Corollary 16 and Theorem 17) by contrasting functions *related by* $\lesssim$, and
- secondly (in Theorem 18) by contrasting functions *related by* $\leq$.

Theorem 15: If $f \lesssim g$, then

- $\mathsf{C}_{\mathcal{R}}(f,A) \subseteq \mathsf{C}_{\mathcal{R}}(g,A)$ and
- $\mathsf{NC}_{\mathcal{R}}(f,A) \subseteq \mathsf{NC}_{\mathcal{R}}(g,A)$.

In particular, if $f \leq g$, then, a fortiori, $f \lesssim g$, and so the conclusions still hold.

Proof: The proof requires only the transitivity of $\lesssim$: if a problem is in $\mathsf{C}_{\mathcal{R}}(f,A)$ by virtue of its being solved by deterministic computer Φ with A $\mathcal{R}$-dominant and $A^*_{\Phi} \lesssim f$, then certainly $A^*_{\Phi} \lesssim g$, and so the problem is in $\mathsf{C}_{\mathcal{R}}(g,A)$.

Similarly, if a problem is in $\mathsf{NC}_{\mathcal{R}}(f,A)$ by virtue of its being solved by non-deterministic computer Ψ with A $\mathcal{R}$-dominant and $A^*_{\Psi} \lesssim f$, then certainly $A^*_{\Psi} \lesssim g$, and so the problem is in $\mathsf{NC}_{\mathcal{R}}(g,A)$. ∎

Consequently, we have the following.

Corollary 16: If $f \lesssim g$ (and so, in particular, if $f \leq g$), then

- $\mathsf{C}_{\mathcal{R}}(f) \subseteq \mathsf{C}_{\mathcal{R}}(g)$ and
- $\mathsf{NC}_{\mathcal{R}}(f) \subseteq \mathsf{NC}_{\mathcal{R}}(g)$.

Proof: By definition, $\mathsf{C}_{\mathcal{R}}(f) = \bigcup_{R\in\mathcal{R}} \mathsf{C}_{\mathcal{R}}(f,R)$. By $|\mathcal{R}|$ evocations of Theorem 15, this is a subset of $\bigcup_{R\in\mathcal{R}} \mathsf{C}_{\mathcal{R}}(g,R)$, which, by definition, is $\mathsf{C}_{\mathcal{R}}(g)$, as required. Similarly NC. ∎

Theorem 17: Let f and g satisfy $f \lesssim g$. Then no relation of class inclusion (neither '$\subseteq$' nor '$\supseteq$') holds in generality between

- $\mathsf{B}_{\mathcal{R}}(f)$ and $\mathsf{B}_{\mathcal{R}}(g)$;

similarly between

- $\mathsf{NB}_{\mathcal{R}}(f)$ and $\mathsf{NB}_{\mathcal{R}}(g)$.

Proof: There exist f and g satisfying $f \lesssim g$, but with $g \leq f$ everywhere; e.g., $f\colon n \mapsto 2n$, $g\colon n \mapsto \left\lfloor \frac{n}{2} \right\rfloor$. The claim of the theorem is witnessed by any computer with an identity time-complexity function $T^*(n) = n$, which necessarily exists by the time-constructibility of polynomials. ∎

However, the following restriction yields a more definite result.

Theorem 18: If $f \leq g$, then

- $\mathsf{B}_{\mathcal{R}}(f) \subseteq \mathsf{B}_{\mathcal{R}}(g)$ and
- $\mathsf{NB}_{\mathcal{R}}(f) \subseteq \mathsf{NB}_{\mathcal{R}}(g)$.

Proof: A problem in $\mathsf{B}_{\mathcal{R}}(f)$ is solved by a deterministic computer Φ with $\mathcal{B}_{\mathcal{R},\Phi} \leq f$. By transitivity of $\leq$, $\mathcal{B}_{\mathcal{R},\Phi} \leq g$, and so the problem is (by virtue of Φ) in $\mathsf{B}_{\mathcal{R}}(g)$.

Similarly, a problem in $\mathsf{NB}_{\mathcal{R}}(f)$ is solved by a non-deterministic computer Ψ with $\mathcal{B}_{\mathcal{R},\Psi} \leq f$. By transitivity of $\leq$, $\mathcal{B}_{\mathcal{R},\Psi} \leq g$, and so the problem is (by virtue of Ψ) in $\mathsf{NB}_{\mathcal{R}}(g)$. ∎

Finally, we consider (in Theorem 19) the dependency of our classes $\mathsf{C}_{\mathcal{R}}(f, A)$ and $\mathsf{NC}_{\mathcal{R}}(f, A)$ upon their *resource parameter* A, by contrasting two resources of which one is always pairwise dominant.

Theorem 19: Suppose that ($\mathcal{R}$-dominant) resources A and B are such that $A^*_\Phi \lesssim B^*_\Phi$ for all computers Φ (notably, this inclusion is certainly true when $A^*_\Phi \leq B^*_\Phi$ for all Φ). Then

- $\mathsf{C}_{\mathcal{R}}(f, B) \subseteq \mathsf{C}_{\mathcal{R}}(f, A)$ and
- $\mathsf{NC}_{\mathcal{R}}(f, B) \subseteq \mathsf{NC}_{\mathcal{R}}(f, A)$.

Proof: A problem in $\mathsf{C}_{\mathcal{R}}(f, B)$ is solved by a deterministic computer Φ with $B^*_\Phi \lesssim f$ and with B $\mathcal{R}$-dominant for Φ. By transitivity of $\lesssim$, then, $A^*_\Phi \lesssim f$, and so (since A is $\mathcal{R}$-dominant) the problem is in $\mathsf{C}_{\mathcal{R}}(f, A)$ by virtue of Φ.

Similarly, a problem in $\mathsf{NC}_{\mathcal{R}}(f, B)$ is solved by a non-deterministic computer Ψ with $B^*_\Psi \lesssim f$ and with B $\mathcal{R}$-dominant for Ψ. By transitivity of $\lesssim$, then, $A^*_\Psi \lesssim f$, and so (since A is $\mathcal{R}$-dominant) the problem is in $\mathsf{NC}_{\mathcal{R}}(f, A)$ by virtue of Ψ. ∎

For a brief summary of the above theorems, see Table I.

TABLE I

A brief summary of the class-inclusion theorems of Sect. III. Where no class inclusion (neither '$\subseteq$' nor '$\supseteq$') holds in generality between classes A and B, we write '$\mathsf{A} \pitchfork \mathsf{B}$'; similarly, where no order (neither '$\lesssim$' nor '$\leq$', in either direction) holds in generality between functions f and g, we write '$f \pitchfork g$'. Numbers in parentheses at the bottom-right of cells indicate the corresponding theorems/corollaries.

	$\mathcal{R} \subseteq \mathcal{S}$	$\mathcal{R} \cap \mathcal{S} = \emptyset$	$f \lesssim g$	$f \leq g$	$A^* \lesssim B^*$
		$\mathcal{R} \nsubseteq \mathcal{S} \nsubseteq \mathcal{R}$, $\mathcal{R} \cap \mathcal{S} \neq \emptyset$			$A^* \leq B^*$
$\mathsf{C}_{\mathcal{R}}(f, A)$	$\mathsf{C}_{\mathcal{S}} \subseteq \mathsf{C}_{\mathcal{R}}$ (10)	$\mathsf{C}_{\mathcal{R}} \pitchfork \mathsf{C}_{\mathcal{S}}$ (13/14)	$\mathsf{C}(f, A) \subseteq \mathsf{C}(g, A)$ (15)	$\mathsf{C}(f, A) \subseteq \mathsf{C}(g, A)$ (15)	$\mathsf{C}(f, B) \subseteq \mathsf{C}(f, A)$ (19)
$\mathsf{NC}_{\mathcal{R}}(f, A)$	$\mathsf{NC}_{\mathcal{S}} \subseteq \mathsf{NC}_{\mathcal{R}}$ (10)	$\mathsf{NC}_{\mathcal{R}} \pitchfork \mathsf{NC}_{\mathcal{S}}$ (13/14)	$\mathsf{NC}(f, A) \subseteq \mathsf{NC}(g, A)$ (15)	$\mathsf{NC}(f, A) \subseteq \mathsf{NC}(g, A)$ (15)	$\mathsf{NC}(f, B) \subseteq \mathsf{NC}(f, A)$ (19)
$\mathsf{C}_{\mathcal{R}}(f)$	$\mathsf{C}_{\mathcal{S}} \subseteq \mathsf{C}_{\mathcal{R}}$ (11)	$\mathsf{C}_{\mathcal{R}} \pitchfork \mathsf{C}_{\mathcal{S}}$ (13/14)	$\mathsf{C}(f) \subseteq \mathsf{C}(g)$ (16)	$\mathsf{C}(f) \subseteq \mathsf{C}(g)$ (16)	(N/A)
$\mathsf{NC}_{\mathcal{R}}(f)$	$\mathsf{NC}_{\mathcal{S}} \subseteq \mathsf{NC}_{\mathcal{R}}$ (11)	$\mathsf{NC}_{\mathcal{R}} \pitchfork \mathsf{NC}_{\mathcal{S}}$ (13/14)	$\mathsf{NC}(f) \subseteq \mathsf{NC}(g)$ (16)	$\mathsf{NC}(f) \subseteq \mathsf{NC}(g)$ (16)	
$\mathsf{B}_{\mathcal{R}}(f)$	$\mathcal{B}_{\mathcal{R}} \lesssim \mathcal{B}_{\mathcal{S}}$, $\mathsf{B}_{\mathcal{R}} \pitchfork \mathsf{B}_{\mathcal{S}}$ (12)	$\mathcal{B}_{\mathcal{R}} \pitchfork \mathcal{B}_{\mathcal{S}}$, $\mathsf{B}_{\mathcal{R}} \pitchfork \mathsf{B}_{\mathcal{S}}$ (13/14)	$\mathsf{B}(f) \pitchfork \mathsf{B}(g)$ (17)	$\mathsf{B}(f) \subseteq \mathsf{B}(g)$ (18)	
$\mathsf{NB}_{\mathcal{R}}(f)$	$\mathcal{B}_{\mathcal{R}} \lesssim \mathcal{B}_{\mathcal{S}}$, $\mathsf{NB}_{\mathcal{R}} \pitchfork \mathsf{NB}_{\mathcal{S}}$ (12)	$\mathcal{B}_{\mathcal{R}} \pitchfork \mathcal{B}_{\mathcal{S}}$, $\mathsf{NB}_{\mathcal{R}} \pitchfork \mathsf{NB}_{\mathcal{S}}$ (13/14)	$\mathsf{NB}(f) \pitchfork \mathsf{NB}(g)$ (17)	$\mathsf{NB}(f) \subseteq \mathsf{NB}(g)$ (18)	

IV. Discussion

Our notion of dominance formalizes resources' relevance to computational processes: resources that are dominant impose the greatest asymptotic cost, so much so that non-dominant resources can be disregarded as irrelevant. Thus, much as the pre-ordering $\lesssim$ can be used to compare the respective time-complexity functions of Turing machines (or similar) and thereby compare their overall efficiency, $\lesssim$ can also be used to compare the respective *$\mathcal{R}$-complexity* functions of *arbitrary-paradigm computers* and thereby compare their overall efficiency.

Furthermore, we have specified (in Definitions 7 and 8) complexity classes in which are categorized problems according to their cost in terms of relevant (i.e., *dominant*) resources. Consequently, we have a framework in which meaningful, consistent comparison of model-heterogenous sets of computers is possible; the framework's complexity classes can accommodate computers conforming to various computational paradigms, and can provide structure reflecting the cost of computation in terms of various resources.

The model-heterogeneity of our framework offers an immediate and important advantage: a *problem's* complexity, which is the most commonly sought object in computational complexity theory, is bounded above by the complexity of the most efficient, known *solution method for the problem*; the ability to compare model-heterogeneous—and, hence, larger—sets of solution methods results in a lower minimal complexity of methods, and, hence, tighter upper bounds on the complexity of problems themselves.

A further advantage (especially for the unconventional-computation community) of the definitions proposed in the present paper is that a newly-designed, non-standard computer that solves a problem can be meaningfully compared with the benchmark of an existing, standard computer that solves the same problem.

We hope that this work, and the above-mentioned advantages thereof, are of interest and use to practitioners of complexity theory and unconventional computing, to readers of the *International Journal of Computers*, and to participants of *ISTASC '10*.

Acknowledgment

We thank Bob Coecke and Joël Ouaknine for their continued support, supervision and suggestions; participants of conferences attended/organized by the author for their encouraging feedback and useful discussion; and reviewers of publications to which the author has contributed for their detailed comments. Finally, we thank the organizers of *ISTASC '10* for their kind invitation to present this work.

References

[1] E. Blakey, "Beyond Blum: what is a resource?" *International Journal of Unconventional Computing*, OCP Science, 2010, to be published.

[2] E. Blakey, "Factorizing RSA keys, an improved analogue solution", *New Generation Computing*, vol. 27, no. 2, Y. Suzuki, M. Hagiya, H. Umeo, and A. Adamatzky (guest editors), Ohmsha/Springer, 2008.

[3] E. Blakey, "On the computational complexity of physical computing systems", *Unconventional Computing 2007*, A. Adamatzky, L. Bull, B. De Lacy Costello, S. Stepney, and C. Teuscher (editors), Luniver Press, 2007.

[4] E. Blakey, "Unconventional complexity measures for unconventional computers", *Natural Computing*, Springer, 2010, to be published.

[5] C. Chang, C. Lin, and J. Lee, "A file protection system based on a trapdoor one-way hash function", *WSEAS International Conference on Applied Computer Science*, Hangzhou, China, 2006.

[6] S. Iriyama and M. Ohya, "On generalized quantum Turing machine and its language classes", *WSEAS International Conference on Applied Mathematics*, Texas, USA, 2007.

[7] S. Negulescu, C. Oprean, C. Kifor, and I. Carabulea, "Elitist ant system for route allocation problem", *WSEAS International Conference on Applied Informatics and Communications*, Rhodes, Greece, 2008.

[8] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1995.

[9] B. Rylander and J. Foster, "Genetic algorithms, and hardness", *WSEAS International Conference on Evolutionary Computation*, Tenerife, Spain, 2001.

[10] A. Sharma, R. Kumar, and P. Grover, "Empirical evaluation and critical review of complexity metrics for software components", *WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, Corfu, Greece, 2007.