A Model-Independent Theory of Computational Complexity: From Patience to Precision and Beyond

Ed Blakey
The Queen's College, Oxford
ed.blakey@queens.oxon.org

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

Submitted in Trinity Term, 2010
for the degree of Doctor of Philosophy in Computer Science

Acknowledgements and Dedication

We thank the author's supervisors, Dr Bob Coecke and Dr Joël Ouaknine, for their continued support and suggestions during this DPhil project. We thank the author's Transfer/Confirmation of Status assessors, Prof. Samson Abramsky and Prof. Peter Jeavons, first for agreeing to act as assessors and secondly for their useful and formative comments about this project; we thank the author's Examiners, Prof. Peter Jeavons and Prof. John Tucker, for taking the time to act as such and for their valuable suggestions, of which some are incorporated here. We thank the organizers of *Unconventional Computing*, the *International Workshop on Natural Computing*, *Quantum Physics and Logic/Development of Computational Models*, *Science and Philosophy of Unconventional Computing* and the *International Conference on Systems Theory and Scientific Computation* for the opportunity (and, in the case of the last-mentioned conference, the kind invitation) to present work forming part of this project. We thank the participants of the above-mentioned conferences/workshops, as well as the *British Colloquium for Theoretical Computer Science* and *Complexity Resources in Physical Computation*, for their encouraging feedback and insightful discussion. We thank the reviewers of publications to which the author has contributed (including *New Generation Computing*, the *International Journal of Unconventional Computing* and *Natural Computing*, as well as proceedings/publications associated with the conferences and workshops mentioned above) for their detailed comments and helpful suggestions; we thank also Prof. José Félix Costa for his kind invitation to contribute to the last-mentioned journal. We thank collaborators and colleagues for showing an interest in this work, for useful discussion and corrections, for bringing to the author's attention many relevant references, and for helping to shape and direct this research (as have the conference participants and publication reviewers—and of course supervisors and assessors—mentioned above); notably, we thank Prof. Cristian Calude for fascinating discussions regarding accelerated Turing machines and the conjectured incompleteness of axiomatizations of resource, Dr Viv Kendon and colleagues (not least Rob Wagner) for the opportunity to explore the interface between the present project and the practical considerations of quantum computation/simulation, Dr Rebecca Palmer for noticing an omission relating to wave superposition, András Salamon for inspiring parts of the present work with questions concerning gap theorems and combined resources, Prof. Susan Stepney for her championing the 'natural' in 'natural computation', and Dr Damien Woods for his suggestions relating to restriction of precision.

Academic assistance aside, we thank the administrative and support staff both at the Oxford University Computing Laboratory and at the Queen's College for their help during the author's DPhil programme; we thank in particular Janet Sadler for her assistance with the organization of the workshop *Complexity Resources in Physical Computation* and with the administration of the below-mentioned EPSRC funding.

We acknowledge the generous financial support of the EPSRC; this work forms part of, and as of 1.x.2008 is funded by, the EPSRC project *Complexity and Decidability in Unconventional Computational Models* (EP/G003017/1).

Personal thanks are due from the author to his family; to his friends in Oxford, Warwick and elsewhere; and to his girlfriend, Gemma.

The author dedicates this dissertation to the memory of his mother, Linda (1944 – 2009).

Abstract

A Model-Independent Theory of Computational Complexity: From Patience to Precision and Beyond

Ed Blakey, The Queen's College

Submitted in Trinity Term, 2010 for the degree of Doctor of Philosophy in Computer Science

The field of computational complexity theory—which chiefly aims to quantify the difficulty encountered when performing calculations—is, in the case of conventional computers, correctly practised and well understood (some important and fundamental open questions notwithstanding); however, such understanding is, we argue, lacking when unconventional paradigms are considered. As an illustration, we present here an analogue computer that performs the task of natural-number factorization using only polynomial time and space; the system's true, exponential complexity, which arises from requirements concerning precision, is overlooked by a traditional, 'time-and-space' approach to complexity theory. Hence, we formulate the thesis that unconventional computers warrant unconventional complexity analysis; the crucial omission from traditional analysis, we suggest, is consideration of relevant resources, these being not only time and space, but also precision, energy, etc.

In the presence of this multitude of resources, however, the task of comparing computers' efficiency (formerly a case merely of comparing time complexity) becomes difficult. We resolve this by introducing a notion of overall complexity, though this transpires to be incompatible with an unrestricted formulation of resource; accordingly, we define normality of resource, and stipulate that considered resources be normal, so as to rectify certain undesirable complexity behaviour. Our concept of overall complexity induces corresponding complexity classes, and we prove theorems concerning, for example, the inclusions therebetween.

Our notions of resource, overall complexity, normality, etc. form a model-independent framework of computational complexity theory, which allows: insightful complexity analysis of unconventional computers; comparison of large, model-heterogeneous sets of computers, and correspondingly improved bounds upon the complexity of problems; assessment of novel, unconventional systems against existing, Turing-machine benchmarks; increased confidence in the difficulty of problems; etc. We apply notions of the framework to existing disputes in the literature, and consider in the context of the framework various fundamental questions concerning the nature of computation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A Tale of Two Complexities

It was the best of times. Computational complexity theory, as it pertains to standard, Turing-machine-like computers, is in many respects well understood. The field has achieved sufficient maturity to guide the practical direction of our algorithm-designing efforts (notably, NP-hardness is routinely taken as excuse enough for our inability to find efficient methods). The field enjoys undeniable success in characterizing problems according to their solutions' cost. The field formalizes and answers important questions concerning problems' comparative difficulty, thereby imbuing with a rich, hierarchical structure the space of problems.

And yet, in some respects, it was the worst of times. The field retains many fundamental, open questions (despite, in one notable case, a million-dollar incentive [47]). The field struggles to reconcile its intention to determine the complexity of *problems* with its ability directly to measure the complexity only of *methods that solve these problems.*

> *The field, created as it was to cater primarily for Turing-like models of computation, fails to capture the true complexity of many non-standard (analogue, chemical, quantum, etc.) computers.*

These misgivings about traditional complexity theory, and especially though not exclusively the last—namely, its inadequacy for capturing some unconventional systems' complexity—, motivate the work described in the present dissertation. More specifically, we describe here a framework of computational complexity that allows quantification and meaningful comparison of computers' efficiency, independently of computational model and of types of resource consumed.

There are, then, two distinct (though not disjoint) approaches to computational complexity: the *traditional*, which, having been developed with the Turing machine very much in mind, considers only those resources such as runtime, memory space, etc. relevant to that model; and the *model-independent*—described and developed here—, which incorporates conventional and unconventional resources alike (*required precision* being a notable example of the latter). Each approach has its own structure of complexity classes: the *traditional* hierarchy has, of course, received much attention and undergone much exploration,

and so is reasonably well understood (fundamental, open questions notwithstanding); that the *new* hierarchy has, in contrast, barely been explored is, to an extent, redressed here.

It is the author's hope that the model-independent framework of complexity theory introduced and investigated in this dissertation is both of use and of interest to members of the computational-complexity and unconventional-computing communities, as well as to a wider readership of computer scientists.

1.2 Background

1.2.1 Preliminaries

We outline now some preliminary notions that recur throughout this dissertation. For some, we present not a full description, but rather one that suffices for the purposes of the present work; some details irrelevant here, then, are omitted. Further detail than is given in this section is deferred to the below-cited works (as well as to the relevant subsequent sections of this dissertation), since it is hoped that the reader's preexisting familiarity with at least some of these notions renders additional exposition here redundant.

Complexity Theory.

Computational complexity theory (or, simply, complexity theory) is the field of mathematics/theoretical computer science that concerns the *difficulty* encountered, or the *cost* incurred, when performing calculations. This cost is encapsulated as the amount of *resource* (typically *time*) consumed by the system during calculation; calculations can then be categorized into *complexity classes* according to their cost. See [37, 103] for an introduction to the field.

Computation.

To *compute* is to perform a calculation, which entails

- acceptance of an *input value* (the *argument* or *instance* of the mathematical function/relation/problem being implemented in the calculation),
- some *processing* of the value (during which the person or system performing the calculation acts in accordance with, for example, an *algorithm*, or the *physical laws* to which the system is subject), and
- production of an *output value* (the *answer* to the problem solved, or the/a *value* of the implemented mathematical function/relation, evaluated with the given input value as its argument).

Computation is the term given either to this *process* of computing, or to the resultant *relation* (often a function) of input-output value pairs (with context determining which meaning is intended). See [103,120] for (much) further detail.

Turing Machine.

The *Turing machine*, introduced in [127], is a formalized, mathematical model of systems that compute. It has

- *storage space*, consisting of a one-dimensional tape—extending unboundedly in one direction—of discrete *cells*, each of which either being blank or bearing a symbol taken from the machine's finite *alphabet*; and
- a *tape head* (positioned at one of the cells), which, in each of the machine's discrete time steps, may write a symbol to its current cell, may move along the tape by up to one cell in either direction, and may switch from the current to a new *internal state* (of which there is a finite set)—the symbol written, direction taken and state adopted depend (via the machine's *transition function*—its 'program') upon only the previous cell's symbol and the head's previous state.

This corresponds to the 'input, processing, output' formulation of computation described in Sect. 1.2.1 *Computation*, in that a Turing machine's tape is, by supposition, preloaded with an encoding of the *input* value, the sequence of state/symbol/position transitions constitutes the *processing* stage, and, if and when the machine halts, the tape's contents encode the *output* value (the machine, and computation, are said to *halt* if and when the machine adopts any of the subset of states deemed to be *halting states*; else, the computation's output value, and sometimes the computation itself, are said to be *undefined*).

Halting Problem.

Depending upon the choice of Turing machine and input value, it is possible though not certain that a Turing-machine computation halts. One may naturally question, then, whether or not[1] a given machine-value pair results in a halting computation; this is the *Halting problem* of [127], and the set of pairs that halt (and only these pairs) is the *Halting set*.

This problem cannot be decided by any Turing machine:

it is proven in [127] that there does not exist a machine that, given an encoding of a machine and an input value, returns (say) '1' if the pair halts and '0' otherwise. Thus arises the question of *computability*, i.e., of whether a given function is implementable by a Turing machine.

Church-Turing Thesis.

For each mathematical model of computers (the Turing machine being one example), there is a corresponding division of mathematical functions into the *computable* and the *uncomputable*.[2] It is prima facie possible that there is much variation between the respective such divisions arising from different models; however, this seems not to be the case: many 'reasonable' formulations of computability, including Turing-computability, recursiveness and expressibility in

[1] Though semantically redundant, the "or not" here stresses an important point: that, in order to decide the Halting problem, a Turing machine must, if a machine-value pair halts, declare that it does so, *and, if the pair does not, declare that it does not.* Whereas, we note below, no Turing machine *decides* the problem in this way, there do exist machines that *accept* the problem—i.e., that, if a machine-value pair halts, declare that it does so, but, if the pair does not, either declare that it does not or declare nothing.

[2] For example, the Turing-machine model places into the former category the *identity function* on input values—by virtue of the machine that halts immediately—, and into the latter the *Halting problem*.

the λ-calculus (see [49, 127]), have been shown to be equivalent, and this notion is widely believed, furthermore, to correspond with *effective computability*—the ability to be computed via the mechanistic following of some finite algorithm. This belief, which is not susceptible to rigorous proof since effective computability is an intuitive rather than a formally defined concept, is known as the *Church-Turing thesis*; see [49, 68].

As is suggested in Sect. 1.2.1 *Complexity Theory*, the division between the computable and the uncomputable may be refined: one may ask of a function not only whether it is computable, but, if it is, then also how *resource-efficiently* it may be computed. The belief (less widely held than the Church-Turing thesis, though still supported by much evidence) that not only computability, but also resource-efficiency, is independent of choice of model of computer is known as the *extended Church-Turing thesis* (more precisely, the thesis does permit efficiency to vary between models, but only *polynomially*[3]); see [57, 135].

Throughout this dissertation, we mention 'standard', 'traditional' and 'algorithmic' computers; by these we mean Turing machines or instances of some model of polynomially equivalent efficiency.

Unconventional Computation.

Perhaps in part because of the Church-Turing thesis and in part because of the close correspondence between Turing machines and the digital computing systems that perform virtually all real-world calculations, computers are typically modelled using the Turing-machine model or equivalent; an according bias exists in dependent fields such as complexity theory. However, many models exist—both as the subject of theoretical studies and (with varying degrees of success/acceptance) as physically realized devices—that do not conform to the standard computational paradigm. These *unconventional computers* include, but are by no means limited to: quantum, chemical, analogue, optical, kinematic and slime-mould computers; as long as a system has provision for accepting input values and—after some form of processing—for supplying output values, then it computes (cf. Sect. 1.2.1 *Computation*).

For example,

- a digital computer's input values may be typed on a keyboard and its output values presented on a screen (with processing—implemented as an algorithmic program, which is in turn implemented electronically—occurring between input and output); and
- a chemical computer's input and output values may be encoded in the concentrations of solutions respectively supplied to and drawn from the system (with chemical reaction occurring between input and output).

More generally, computers accept input values encoded in the values of *manipulable parameters* of the system, and supply output values encoded in the values of *mensurable parameters* of the system (with processing, in the form of the system's evolving in accordance with the laws—which may, for example, be

[3] If a reasonable model implements a function f such that the required resource is a function $R(n)$ of the size n of the input value, then the thesis implies that any other reasonable model $\mathcal{M}$ can compute f using resource bounded above by some polynomial—depending upon only $\mathcal{M}$ and f, and, in particular, certainly not upon n—in $R(n)$.

electrical, chemical or quantum mechanical—to which it is subject, occurring between input and output).

This model-independent view of computation—which accommodates not only standard computers, but also more natural, physical systems—may allow new approaches to traditionally difficult problems. As we argue in this dissertation, however, part of the cost of this freedom is the need for innovative forms of complexity analysis.

Aside. Often, an unconventional system will lend itself naturally to a particular computational process, whilst being unsuitable for more general tasks.[4] In particular, it may be more viable to have (say) a Turing machine convert the user's input value into a form processable by the unconventional system and convert the unconventional system's output into a user-readable form, than to have the unconventional system operate unaided by such a Turing-machine 'harness'. Such hybrid systems appear throughout the present work; see, for example, Footnote 14 of Chap. 2 and Remark 16.

See [2, 78] for discussion of the field of unconventional computing as well as its importance and timeliness.

Resource.

In order to quantify the efficiency of a computing device, one measures the *resource* consumed during its operation. For Turing machines, the resources typically considered are *time* and *space*—respectively a computation's *duration* measured in time steps and its *memory usage* in tape cells. When (as here) the notion of computer is widened so as to include unconventional paradigms, then the notion of resource may—and, in some cases, does—need according extension; resources consumed by—and that should be considered during complexity analysis of—unconventional computers include *energy* and *precision*, for example.[5] See [23, 51] for more on resources.

Slightly more formally, we may model resources as functions (dependent upon the *computing system* in question) that map each *input value* to the corresponding *amount* of the resource consumed by the computer in processing that value. This notion is developed in Chaps. 3 and 4.

Blum's Axioms.

Blum's axioms, introduced and investigated in [31], stipulate of a resource

1. that it be defined at those inputs—and at those inputs only—at which the computation being measured is defined, and

2. that it be a computable problem to determine whether a given value is indeed the amount of the resource consumed in processing a given input.

[4] This unsuitability is in part because certain unconventional paradigms are not easily programmable in a high-level way—practitioners of such paradigms may perform manipulations at a low level, observing and exploiting behaviour exhibited by their systems, whilst lacking a library/language of high-level commands with which to achieve arbitrary computational needs; see the aside of Sect. 3.6.2 for a recognition of this situation in the context of *quantum* computation.

[5] That 'resource' needs extension to the non-standard is an important observation, since the availability of unconventional resources may impinge on a system's efficiency long before availability of time or space has become pressing.

The axioms are in many contexts desirable (a notable exception is when resources are allowed to be *non-deterministic*, which transpires to conflict with the latter axiom), and so the notion of resource is often taken to include them. See Sect. 3.4.1 *Blum's Axioms* for further detail.

Complexity Function.

Whereas resource functions map each input *value* to the amount of resource required to process that value, *complexity functions* (see Sect. 3.4.3) map each input *size* to the minimal amount of resource sufficient to process any input value of that size. This encapsulates the way in which resources *scale* as the input value grows, and complexity theorists are particularly interested in whether this scaling is logarithmic, polynomial (and of what degree), exponential, etc. See also [1, 37, 103].

Asymptotic Notation.

The use of *asymptotic*, especially $\mathcal{O}$-, *notation* (discussed in [103], amongst many others) is widespread in complexity theory. It provides a suitable language with which to capture the *large-scale behaviour* of complexity functions, allowing abstraction and isolation of their complexity-theoretically relevant properties: whether (and of what degree) the functions are polynomial, etc.

To say that an algorithm requires, say, $6n^2 - 2n + 10$ milliseconds to process an input value of size n is to make an implicit assumption about the *implementation* of the algorithm (that it uses a processor of a specific speed, for example); a better processor may execute the same algorithm in $3n^2 - n + 5$ milliseconds. Since it is typically the time complexity of the *algorithm* itself—and, ultimately, of the *problem* that the algorithm solves—, rather than of any of the algorithm's physical implementations (with specific-speed processors and so on), that one wishes to measure (see Sect. 4.1.1 *Complexity: Problems versus Solution Methods*), the relevant information is that the time complexity is *quadratic* in n; more exactly, the complexity theorist is often interested only in the fact that, *but for a finite number (or bounded set) of exceptional values of* n—which necessarily all occur where n is less than some finite threshold—, the time complexity behaves quadratically. This is precisely the flavour of condition captured by $\mathcal{O}$-notation, as is evident from the following definition.[6]

Definition 1. Let $\mathcal{O}(g(n))$ denote the class of all functions $f(n)$ such that there exist a *threshold* n_0 and *constant* c such that, for all $n > n_0$, $|f(n)| \leq c\,|g(n)|$.

In many of the uses of this notation in the present work, f and g are functions mapping natural numbers to natural numbers.

Aside. Returning to the quadratic-time algorithm above, then, we see that the respective time-complexity functions of $6n^2 - 2n + 10$ ms and $3n^2 - n + 5$ ms are

[6] In addition to the notation's ignoring arbitrary, irrelevant details such as processor speed, its appropriateness in the context of complexity theory is further bolstered by speed-up theorems, etc. For example, for any algorithm with time complexity $f(n)$ and any real number $\epsilon > 0$, there exists an equivalent algorithm with time complexity $\epsilon f(n) + n + 2$ (see Sect. 2.4 of [103]); hence, we may disregard as irrelevant all but the *degree* of a polynomial time-complexity function—coefficients, in particular, may be ignored.

both in $\mathcal{O}\left(n^2\right)$: letting c (as in Definition 1) be 6 ms and n_0 be 5, the definition is satisfied for either processor. Neither can we improve upon this, in the sense that there is no choice of c and n_0 that demonstrates that these complexity functions are in $\mathcal{O}\left(n^{2-\epsilon}\right)$ for any positive real number ϵ (we leave the details to the reader, claiming only that any $n > \max\left\{n_0, \sqrt[\epsilon]{\frac{c}{2}}\right\}$ offers a counterexample); that no such improvement can be made may be formalized with other forms of asymptotic notation, though we use only $\mathcal{O}$ in the present dissertation.

We state and prove now a lemma concerning $\mathcal{O}$-notation (and, in particular, its distributive interaction with addition) that we evoke in the proof of Theorem 9.

Lemma 1. For non-negative functions f_i and g_i, if $f_i \in \mathcal{O}\left(g_i\right)$ for all $i \in \{1, 2, 3, \ldots, k\}$, then $\sum_i f_i \in \mathcal{O}\left(\sum_i g_i\right)$.

Proof. For each i, we have that $f_i \in \mathcal{O}\left(g_i\right)$, whence there exist a threshold n_i and constant c_i such that $\left|f_i\left(n\right)\right| \leq c_i \left|g_i\left(n\right)\right|$ for all $n > n_i$. Letting $n_0 = \max\left\{n_1, n_2, n_3, \ldots, n_k\right\}$ and $c = \max\left\{c_1, c_2, c_3, \ldots, c_k\right\}$, then, we have that, for all $n > n_0$,

$$\begin{aligned}
\left|\sum_i f_i\left(n\right)\right| &\overset{\text{Non-negativity of } f_i}{=} \sum_i \left|f_i\left(n\right)\right| \\
&\leq \sum_i c_i \left|g_i\left(n\right)\right| \\
&\leq \sum_i c \left|g_i\left(n\right)\right| \\
&\overset{\text{Non-negativity of } g_i}{=} c\left|\sum_i g_i\left(n\right)\right| \quad ,
\end{aligned}$$

as required. □

Natural Numbers.

In Chap. 2, and Chap. 2 alone, we deem zero not to be a natural number: $\mathbb{N} := \{1, 2, 3, \ldots\}$. Throughout the rest of this dissertation, we take zero to be natural: $\mathbb{N} := \{0, 1, 2, \ldots\}$; as an important consequence, certain definitions of Chap. 3 allow resource and complexity functions to take the value zero.

1.2.2 Literature

The existing literature that forms part of the background to the present project is cited throughout this dissertation and listed in the bibliography (which begins on p. 200); notably, we relate the project to certain of the bibliographical items in Sects. 3.1.1, 3.1.2 and 3.3.4.

1.3 Overview

We give now a brief overview of the content of this dissertation.

In Chap. 2 (***Motivation***), we consider the mathematical problem of *factorization*[7]. We describe two analogue-computer systems that factorize natural numbers, and analyze[8] the systems so as to demonstrate that they operate with *polynomial* time and space complexity. However, whereas this seemingly renders the systems more efficient than known, conventional-computer methods of factorizing, we demonstrate also that the system's *precision complexity* (outlined in Chap. 2 and formalized in Chap. 3) is *exponential*, and, hence, prohibitive of the systems' efficient use. The implication is that successful complexity analyses of unconventional computers sometimes require consideration of unconventional resources (for, as in the factorization example, a 'standard' complexity analysis may overlook the true complexity of a system); thus is motivated the *model-independent framework of computational complexity theory* presented in this dissertation. (An additional, practical purpose of Chap. 2 is to illustrate the derivation and analysis of computers as prescribed by the model-independent framework.)

In Chap. 3 (***Resource***), we begin to describe the framework. After a discussion of the computational resource of *precision* (encountered in Chap. 2 in the context of the factorization systems), we abstract the crucial properties so as to begin formalization of a general notion of *resource*. We discuss also specific examples of resources, pertaining to several different computational paradigms.

In Chap. 4 (***Dominance***), we continue to describe our complexity framework. We deal, in particular, with the issue of *comparison* of computers' efficiency, which ability is implemented using a notion of computers' *overall complexity*, which, in turn, is defined in terms of *dominance*, a criterion that formalizes resources' 'relevance' to a computation (to dominance correspond *complexity classes*, which we introduce and investigate—via inclusion theorems, etc.—in Chap. 4). However, having introduced a means by which computers' efficiency may be compared, we find an inconsistency between the notions behind this means and certain resources (which, admittedly, are contrived, though are certainly valid according to the description of Chap. 3): without further restriction, the formalization of resource begun in Chap. 3, when used in conjunction with dominance and related notions, leads to undesirable complexity behaviour. We introduce *normality* (a property of resources), and stipulate that the resources with which we work be *normal*, in order to eliminate some of this unwanted behaviour.

The ideas introduced in Chaps. 3 and 4 are central and fundamental to the present work's model-independent framework of complexity theory; they constitute the chief theoretical contribution of this DPhil project.

In Chap. 5 (***Case Studies***), we apply our framework's theoretical concepts —notably *precision*—to several practical case studies, with a view to resolving controversial aspects thereof; we consider also the extent to which this resolution is carried out.

In the sixth and final chapter (***Discussion***), we consider the analogues sug-

[7]For clarity and unambiguity, we use in this dissertation the verb 'factorize' rather than 'factor' (and, hence, 'factorization' instead of 'factoring', etc.), reserving 'factor' as a noun.

[8]We spell the verb 'analyze' with a 'z' rather than an 's' (even though in the case of other words we adhere to British rather than United States spelling conventions) because we wish, so as to be clear and unambiguous, to be able to distinguish this verb's third-person, singular, indicative, present-tense form ('analyzes') from the plural form ('analyses') of the noun 'analysis'.

gested by our framework of such concepts as cryptography, Kolmogorov complexity and set theory. Further, we view within the context of the framework various issues surrounding the fundamental nature of computation: complexity's inherence in computational tasks (rather than in computers), the delimiting of a computer from its environment, the existence of an underlying computational resource, and so on. We gather ideas (from throughout this dissertation) for future extension of the present project, and make some concluding comments.

Whereas Chaps. 3 and 4 represent the majority of the present project's *theoretical* contribution, Chap. 5 offers its chief *practical* application, and Chap. 6 its relationship to *foundational* issues.

Chapter 2

Motivation

2.1 Introduction

An important and motivating thesis of the present project is that

unconventional computers warrant unconventional complexity analysis.

That is to say that, in analyzing the computational complexity of an unconventional computer, one cannot apply the techniques of traditional, Turing-machine-type complexity theory and realistically expect the analysis thereby performed to be necessarily accurate, meaningful or correct. The standard tools of complexity theory are by definition suitable for analyzing the standard computers for which they were designed, but there exist, we claim, non-standard computers that demand markedly different complexity analyses.

In order to justify the above thesis, we discuss in this chapter a computing system (and an improvement thereon) of which the true computational complexity is not captured by traditional complexity analysis. Specifically, the system is an analogue/optical computer that factorizes natural numbers[1] (we describe this task below—see Sect. 2.1.1); analysis following the pattern of traditional complexity theory suggests that the system has *polynomial* time and space complexity (which, indeed, it does), but we argue that resources other than time and space are consumed by the system, and that this consumption scales *exponentially*—we argue, in short, that standard complexity analysis overlooks the true complexity of the system. More specifically than the above thesis, then, the content of the present chapter highlights that the failure of conventional complexity theory to cater for unconventional computers is an issue of *resource*.

Additionally to the motivational intent of Chap. 2, the chapter serves a practical purpose: its content offers detailed worked examples of the way in which computational systems can be derived and analyzed, from the identification of a *problem* of interest (in our case, the notoriously difficult factorization problem), via both development of a *strategy* with which to tackle the problem and implementation thereof as a *physical system* (we formulate the problem geometrically with a view to direct, physical instantiation), to *complexity analysis*—with respect to all relevant resources—of the system; we hope that this

[1] In this chapter, we deem 0 not to be a natural number: we define the set $\mathbb{N}$ of natural numbers to be $\{1, 2, 3, \ldots\}$.

exposition is of use to practitioners of (especially unconventional) computing when designing/reasoning about their computers.

Note that the detail with which we analyze the complexity of the systems of the present chapter is necessary in that this detail gives rise to the rigour that allows us to state (rather than merely to suggest) that there exist systems with time and space complexity *but not precision complexity* suggestive of tractability (from which statement one may derive the thesis presented at the start of this chapter). This is not to say that the detail is necessary in order broadly to *understand* the main, motivational argument of Chap. 2, and in particular the proofs of the propositions, lemmata, etc. of Sects. 2.2 and 2.3 can safely be disregarded if broad though not rigorous understanding is the reader's aim. This notwithstanding, the detail may also be of instructive use in contributing to the practical purpose mentioned in the previous paragraph, especially when the reader's systems bear similarity in some sense to those of the present chapter. Furthermore, our detailed complexity analysis answers the question of where exactly the 'cheating' occurs—a perfectly natural and tantalizing question given the existence of polynomial-time, polynomial-space factorization systems, and one, therefore, that cannot be ignored.

As a final comment before exploring in detail the present chapter's systems, we note that they arguably offer a less clear demonstration of the importance to complexity analyses of *precision* than do the systems of Sect. 3.3.2, and offer less familiarity to the reader than a similar analysis of, say, the soap-bubble computer of [97]; they are nonetheless included here because of the notorious difficulty of the factorization problem and the historical fact that the former of the two systems of this chapter motivates the present DPhil project.

2.1.1 Factorization

The problem of factorization is simple to state—it is that of identifying the factors of a given natural number—, but, despite being an extremely natural and well- and long-studied problem, has thus far resisted efficient solution.[2] Neither does the lack of such solution stem from a lack of motivation; not only would an efficient factorization method be of great academic interest, it would also have profound consequences for, amongst others, cryptography and the practical applications (internet security, etc.) thereof. We now introduce the problem more formally in two closely related forms.

The Problems, Function and Decision.

The *function* problem of factorization (which we informally outline in the preceding paragraph) is to return the multiset[3] of prime factors of a given natural

[2]This may seem counterintuitive. Our everyday experience tells us that it is easier to take things apart than to put them back together; thermodynamicists, with their inexorable increase in entropy (at least in closed systems yet to achieve equilibrium), would only agree. Yet the 'putting together' of natural numbers via multiplication of their prime factors seems to offer an exception: whilst, given natural numbers a and b, we can efficiently combine them into their product ab (e.g., via long multiplication, which takes time quadratic and space linear in the numbers of digits of a and b), it appears to be much more difficult to 'take apart' ab so as to retrieve a and b.

[3]A *multiset* is similar to a set, but members may be included more than once; whereas 6, 12 and 36 have the same *set* (namely, $\{2, 3\}$) of prime factors, for example, their *multisets*

number; the corresponding *decision* problem asks whether a given natural number has a non-trivial[4] factor less than a given threshold. Formally, then, the problems are as follows:

FACTORIZATION (FUNCTION-PROBLEM VERSION)	
Instance:	$n \in \mathbb{N}$.
Question:	What (as a multiset) are the prime factors of n?

The size of an instance n (which instance we suppose to be presented in standard place notation, say with base ten) is the number $\lfloor \log_{10} n \rfloor + 1$ of digits of n. We may, for complexity-theoretic purposes, use the approximation $\log_{10} n$ to this number.[5]

For example, the instance 90 (which has size 2) leads to the answer $\{2, 3, 3, 5\}$ since $90 = 2 \times 3^2 \times 5$, and the instance $2^{17} - 1$ ($= 131\,071$, size 6) to the answer $\{2^{17} - 1\}$ since $2^{17} - 1$ is prime.

FACTORIZATION (DECISION-PROBLEM VERSION)	
Instance:	$(n, l) \in \mathbb{N}^2$, with $l \leq n$.
Question:	Does there exist a factor a of n such that $1 < a < l$?

The size of an instance (n, l) (the coordinates of which we suppose to be presented in standard place notation, say with base ten) is the number $\lfloor \log_{10} n \rfloor + \lfloor \log_{10} l \rfloor + 2$ of digits of n and l combined. We may, for complexity-theoretic purposes, use the approximation $\log n$ to this number (this logarithm is a factor of between one and two away from the sum of the respective logarithms of n and l: $1 \leq l \leq n$, whence $0 \leq \log l \leq \log n$, whence $\log n \leq \log n + \log l \leq 2 \log n$).

For example, the instance $(90, 3)$ (size 3) yields the answer 'yes' (since $2|90$ and $2 < 3$), and the instance $\left(2^{17} - 1, 2^{17} - 1\right)$ (size 12) the answer 'no' (since $2^{17} - 1$ is prime and so has no non-trivial factors).

The Problems' Equivalence.

The two problem formulations are equally difficult in the sense that a method for solving either one yields an 'essentially equally efficient' method for solving the other, as we now make precise.

Proposition 1. From a method for solving FACTORIZATION (FUNCTION) can be constructed a closely related method for solving FACTORIZATION (DECISION). The methods' respective time-complexity functions differ only by a term that is quadratic in the input size; similarly their space-complexity functions.

($\{2, 3\}$, $\{2, 2, 3\}$ and $\{2, 2, 3, 3\}$ respectively) of prime factors differ. By uniqueness of prime factorization and by accommodation in multisets of repeated members, there exists a bijection between natural numbers and their prime-factor multisets.

[4] The *trivial* factors of $n \in \mathbb{N}$ are 1 and n; all other factors, and only the other factors, are *non-trivial*.

[5] We may simply write $\log n$, rather than $\log_{10} n$, here. When using $\mathcal{O}$-notation, the base to which logarithms are taken does not matter: a change of base is equivalent to multiplication by a constant (which is ignored by $\mathcal{O}$), since $\log_u x = \log_u v \log_v x$.

Informally, the method (for FACTORIZATION (DECISION)) that we construct merely calls the existing method (for FACTORIZATION (FUNCTION)) and checks the resultant multiset for members that are less than the given limit (namely, l in the decision problem's definition).

Proof. Suppose that the method[6] M solves the function version of the problem with time complexity T^*_M and space complexity S^*_M (that is, M needs at most $T^*_M(k)$ units of time (e.g., time steps if M is a Turing machine) and $S^*_M(k)$ units of space (e.g., tape cells if M is a Turing machine) to process an arbitrary input value of size k).

Consider the following method M':

M'	
1.	Accept input value $(n, l) \in \mathbb{N}^2$.
2.	Let P be the multiset returned by method M given input n.
3.	If the minimal element p_0 of P satisfies $p_0 < l$,
4.	then return '**yes**';
5.	else, return '**no**'.

This method solves FACTORIZATION (DECISION) since,

- if the answer to FACTORIZATION (DECISION) given input n is 'yes'—that is, if n has any factor a such that $1 < a < l$—, then n certainly has a *prime* factor p such that $1 < p < l$ (simply let p be any prime factor of a), and so the *minimal* prime factor p_0 of n satisfies $1 < p_0 \leq p < l$, whence M' returns 'yes'; and,

- if the answer to FACTORIZATION (DECISION) given input n is 'no'—that is, if n has no factor a such that $1 < a < l$—, then in particular the minimal prime factor p_0 of n does not satisfy $1 < p_0 < l$ (and, of course, it is not the former inequality $1 < p_0$ but the latter $p_0 < l$ that fails, since all primes are greater than 1), whence M' returns 'no'.

We consider now the time and space complexity of each line of M', supposing that n has ν digits and that l has λ.

- **Line 1**, in which the two place-notation natural numbers n and l are accepted as input, takes time and space *linear* in the numbers' combined length $\nu + \lambda$; this assumes a time overhead for reading digits, and a space overhead for storing digits, that are constant—i.e., these costs do not, in particular, depend upon the position of the digit being read.

- **Line 2**, in which the multiset P of prime factors of n is found (using method M) and stored, takes time $T^*_M(\nu)$, and uses space $S^*_M(\nu)$ (in which space M can be executed and, hence, P calculated) plus $\nu \log_2 n$ digits' space (in which P may be stored, since P has at most $\log_2 n$ members

[6]The method may be an algorithm, a Turing machine implementing such an algorithm, or a digital computer implementing such a machine; alternatively, it may for example be an analogue, chemical or quantum system. The model of computation is not important here, provided that it is endowed with notions of time and space complexity (and virtually all models are indeed so endowed—see Sect. 3.5.1 *Time and Space*).

(since each is at least 2 and their product is n, whence $n = \prod_{p \in P} p \geq \prod_{p \in P} 2 = 2^{|P|}$ and so $\log_2 n \geq |P|$) of which each has at most ν digits).

- **Line 3**, in which the minimal member p_0 of P is identified and compared with l, may consist of $|P|$ comparisons (each, between two numbers consisting of $\mathcal{O}(\nu)$ digits, taking $\mathcal{O}(\nu)$ time and $\mathcal{O}(1)$ space (which latter can be reused for subsequent such comparisons))—the first $|P| - 1$ to identify p_0, the last one to compare p_0 with l—during which comparisons the smallest member of P yet considered is stored, requiring $\mathcal{O}(\nu)$ space. Again since $|P| \leq \log_2 n$, this line takes time in $\mathcal{O}(\nu^2)$; it takes space in $\mathcal{O}(\nu)$.
- **Lines 4** and **5** each take *constant* time and *zero* space.

We summarize this analysis in Table 2.1.

Line:	Time complexity:	Space complexity:
1.	$\mathcal{O}(\nu + \lambda)$	$\mathcal{O}(\nu + \lambda)$
2.	$T^*_M(\nu)$	$S^*_M(\nu) + \mathcal{O}(\nu^2)$
3.	$\mathcal{O}(\nu^2)$	$\mathcal{O}(\nu)$
4.	$\mathcal{O}(1)$	$\mathcal{O}(1)$
5.	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Total	$T^*_M(\nu) + \mathcal{O}(\nu^2 + \lambda)$	$S^*_M(\nu) + \mathcal{O}(\nu^2 + \lambda)$

Table 2.1: A summary of the line-by-line and overall time and space complexity of method M'.

Hence, the overall[7] time and space complexity of M' differs from that of M only by terms quadratic in the size of the input; this is as claimed. □

A specific corollary of Prop. 1 is that a method M for solving FACTORIZATION (FUNCTION) in polynomial (degree $k \geq 2$) time/space leads to a method M' for solving FACTORIZATION (DECISION) also in polynomial (of degree k) time/space; another corollary is that a method M that solves FACTORIZATION (FUNCTION) in exponential time/space leads to M' that solves FACTORIZATION (DECISION) in the same.

Proposition 2. Conversely to Prop. 1, from a method for solving FACTORIZATION (DECISION) can be constructed a closely related method for solving FACTORIZATION (FUNCTION). The methods' respective time-complexity functions differ only by a quadratic multiplicative term and a cubic additive term; similarly their space-complexity functions.

Informally, the method (for FACTORIZATION (FUNCTION)) that we construct repeatedly finds via binary search the least, as-yet-unfound, prime factor of the input value n. The existing method (for FACTORIZATION (DECISION)) is called at each stage of the search so as to determine in which half of the search-space range (initially $\{2, 3, 4, \ldots, n-1\}$) the sought factor lies.

[7] Note that execution of M' follows either the path consisting of lines '1, 2, 3, 4' (if n has a non-trivial factor less than l) or the path '1, 2, 3, 5' (otherwise); the overall complexity stated reflects this.

Proof. Suppose that the method[8] N solves the decision version of the problem with time complexity T_N^* and space complexity S_N^*.

Consider the following method N':

N'	
1.	Accept input value $n \in \mathbb{N}$.
2.	If $n = 1$,
3.	then return $\{\}$.
4.	If, given input (n, n), N returns 'no', *(i.e., 'If n is prime,')*
5.	then return $\{n\}$.
6.	Let $r_{\text{low}} = 2$.
7.	Let $r_{\text{high}} = n - 1$.
8.	Repeat:
9.	If $r_{\text{low}} = r_{\text{high}}$,
10.	then return $\{r_{\text{low}}\} \cup N'\left(\frac{n}{r_{\text{low}}}\right)$.
11.	Let $c = \left\lceil \frac{r_{\text{low}} + r_{\text{high}}}{2} \right\rceil$.
12.	If, given input (n, c), N returns 'yes',
13.	then set r_{high} to $c - 1$;
14.	else, set r_{low} to c.
15.	End 'repeat'. *(i.e., 'Go to line 8.')*

Note that, at line 10, we use the notation '$N'(x)$' to mean the multiset returned by N' given input x.

The intuitive idea of method N' is to find via binary search the least prime factor p of n and recursively to call the method with input $\frac{n}{p}$ in order to find the remaining prime factors.

We defer to the lemmata below the proofs of the following statements, which, together, establish Prop. 2.

- N' solves FACTORIZATION (FUNCTION) (Lemma 2).
- N' has time complexity in $\mathcal{O}\left(\nu^2 T_N^*(\nu) + \nu^3\right)$ and space complexity in $\mathcal{O}\left(\nu^2 S_N^*(\nu) + \nu^3\right)$ (Lemma 3). □

A specific corollary of Prop. 2 is that, if N solves FACTORIZATION (DECISION) in polynomial (say, degree-k) time/space, then N' solves FACTORIZATION (FUNCTION) in polynomial (degree-$(\max\{k+2, 3\})$) time/space; another corollary is that, if N requires exponential time/space, then so does N'.

Lemma 2. N' solves FACTORIZATION (FUNCTION).

Proof. First, note that N' solves FACTORIZATION (FUNCTION) in the special cases '$n = 1$'[9] and 'n is prime' by explicit construction at lines 2/3 and 4/5 respectively.

[8] We use 'method' as in the previous proposition—see Footnote 6.

[9] Note that the multiset of prime factors of 1 is empty—cf. line 3; this is uniquely consistent with rules such as 'X is the multiset of prime factors of k if and only if $X \cup \{q\}$ is the multiset of prime factors of qk, for prime q'. Intuitively, much as one expects the *total* of no numbers to be 0 (the *additive* identity), it is equally sensible to take as the *product* of no numbers the *multiplicative* identity 1.

In other cases (explicitly, when n is composite), N' initializes r_{low} to 2 and r_{high} to $n-1$; writing p for the least prime factor of n, this initialization establishes the invariant

$$r_{\text{low}} \leq p \leq r_{\text{high}} \ . \tag{2.1}$$

We claim that this invariant is maintained throughout execution of N'.

After initialization at lines 6 and 7, variables r_{low} and r_{high} are altered only at lines 13 and 14.

Suppose that invariant (2.1) holds just before execution of line 13. By the conditional statement at line 12 and hypothesizing that line 13 is to be executed, we have that N returns 'yes' given input (n, c), i.e., that n has a factor a in the range $\{2, 3, 4, \ldots, c-1\}$. Clearly $a \geq p$ (by definition of p as the least prime factor of n), whence $p \leq a \leq c-1$, so, in reducing r_{high} to $c-1$ (and leaving r_{low} unchanged), execution of line 13 maintains invariant (2.1).

Similarly, if line 14 is about to be executed (and supposing that invariant (2.1) holds), then we have from the conditional statement at line 12 that N returns 'no' given input (n, c), i.e., that n has no factors in $\{2, 3, 4, \ldots, c-1\}$; in particular, $p \notin \{2, 3, 4, \ldots, c-1\}$. So $p \geq c$, and, in increasing r_{low} to c (and leaving r_{high} unchanged), execution of line 14 maintains the invariant.

Therefore invariant (2.1) is maintained throughout execution of N'.

Consider now the value $d := r_{\text{high}} - r_{\text{low}}$; after initialization, $d = n-3$. In each repetition of the 'repeat' block (lines 8 – 15) such that $d \neq 0$ (i.e., $r_{\text{high}} \neq r_{\text{low}}$, whence line 10 is not executed), exactly one of lines 13 and 14 is executed: either r_{high} becomes $c-1$ or r_{low} becomes c, where $c = \left\lceil \frac{r_{\text{low}} + r_{\text{high}}}{2} \right\rceil$. Note that, since $r_{\text{low}} \in \mathbb{N}$,

$$\left\lceil r_{\text{low}} + \frac{1}{2} \right\rceil = r_{\text{low}} + 1 \ ; \tag{2.2}$$

similarly, since $r_{\text{high}} \in \mathbb{N}$,

$$\left\lceil r_{\text{high}} - \frac{1}{2} \right\rceil = r_{\text{high}} \ . \tag{2.3}$$

Note also that, since non-execution of line 10 together with invariant (2.1) gives that $r_{\text{low}} < r_{\text{high}}$, and since r_{low} and r_{high} are integers, we have that

$$r_{\text{low}} \leq r_{\text{high}} - 1 \ . \tag{2.4}$$

So

$$\begin{aligned} r_{\text{low}} &\overset{(2.2)}{<} \left\lceil r_{\text{low}} + \frac{1}{2} \right\rceil \\ &= \left\lceil \frac{2r_{\text{low}} + 1}{2} \right\rceil \\ &\overset{(2.4)}{\leq} \left\lceil \frac{r_{\text{low}} + r_{\text{high}}}{2} \right\rceil \\ &= c \ , \end{aligned}$$

and

$$\begin{aligned} r_{\text{high}} &\overset{(2.3)}{=} \left\lceil r_{\text{high}} - \frac{1}{2} \right\rceil \\ &= \left\lceil \frac{2r_{\text{high}} - 1}{2} \right\rceil \\ &\overset{(2.4)}{\geq} \left\lceil \frac{r_{\text{low}} + r_{\text{high}}}{2} \right\rceil \\ &= c \\ &> c - 1 \ ; \end{aligned}$$

so $r_{\text{low}} < c$ and $r_{\text{high}} > c - 1$—altering r_{high} to $c - 1$ represents a strict decrease in r_{high}, and altering r_{low} to c represents a strict increase in r_{low}; altering one of r_{high} and r_{low} in this way and leaving the other unchanged (as happens in each $d \neq 0$ repetition of the loop) therefore strictly decreases $d = r_{\text{high}} - r_{\text{low}}$.

Since $d \in \mathbb{Z}$ strictly decreases (from a finite initial value) in each of the loop's repetitions, but is nonetheless bounded below by 0 (by invariant (2.1)), we have that N' must halt. This happens at line 10 when $r_{\text{low}} = r_{\text{high}}$, at which point we have found the least prime factor $p = r_{\text{low}} = r_{\text{high}}$ of n (recall invariant (2.1)) and can recursively call the method with argument $\frac{n}{p}$ to find the remaining prime factors.

Once the penultimate prime factor has been so found, the recursive call is with prime argument, whence we have termination at line 5.

Thus, the method solves FACTORIZATION (FUNCTION). □

Lemma 3. The time complexity $T^*_{N'}$ of N' is in $\mathcal{O}\left(\nu^2 T^*_N(\nu) + \nu^3\right)$. The space complexity $S^*_{N'}$ of N' is in $\mathcal{O}\left(\nu^2 S^*_N(\nu) + \nu^3\right)$.

Proof. We first analyze the time and space complexity of each line of N', supposing that n has ν digits.

- **Line 1**, in which the place-notation natural number n is accepted as input, takes time and space *linear* in ν; this assumes a time overhead for reading digits, and a space overhead for storing digits, that are constant—i.e., these costs do not, in particular, depend upon the position of the digit being read.

- **Line 2**, in which n is compared for equality with 1, takes *constant* time and space (we check in constant time and space whether the first digit of n is 1; if so, we check, again in constant time and space, whether there are no further digits).

- **Line 3**, in which the empty multiset $\{\}$ is returned, takes *constant* time and space.

- **Line 4**, in which N is called with argument (n, n) of size ν (recall from Sect. 2.1.1 *The Problems, Function and Decision* that we may take as the size of argument (n, n) the value $\log n$), takes time $T^*_N(\nu)$ and space $S^*_N(\nu)$.

- **Line 5**, in which multiset $\{n\}$ is returned, takes time and space *linear* in ν, for similar reasons to line 1.

- **Line 6**, in which a variable is assigned the value 2, takes *constant* time and space.
- **Line 7**, in which a variable is assigned the value $n-1$, takes time and space *linear* in ν, again for similar reasons to line 1.
- **Lines 8** and **15**, the markers for the start and end of the method's 'repeat' loop, have no time/space overheads (apart from, say, a constant-space record of line 8's memory address or similar).
- **Line 9**, in which r_{low} and r_{high} (both of $\mathcal{O}(\nu)$ digits) are compared for equality, takes time $\mathcal{O}(\nu)$ and space $\mathcal{O}(1)$; cf. line 3 of M' above.
- **Line 10** returns the partial answer $\{r_{\text{low}}\}$ (in time and space *linear*—recall the analysis of line 5—in $\log r_{\text{low}} \in \mathcal{O}(\nu)$) and calls method N' with input $\frac{n}{r_{\text{low}}}$ (in time at most $T^*_{N'}(\nu)$ and space at most $S^*_{N'}(\nu)$, since it can require no more time/space for N' to factorize m than to factorize qm where q is the least prime factor of qm, because the method's respective executions differ only in factor q's having to be found in the latter instance).
- **Line 11**, in which a variable is assigned the value $\left\lceil \frac{r_{\text{low}}+r_{\text{high}}}{2} \right\rceil$ where $r_{\text{low}}, r_{\text{high}} \in \mathcal{O}(n)$, takes time and space *linear* in ν (notably, digit-by-digit addition requires linear time and space).
- **Line 12**, in which N is called with argument (n, c) (where $c \leq r_{\text{high}} < n$) of size ν, takes time at most $T^*_N(\nu)$ and space at most $S^*_N(\nu)$.
- **Lines 13** and **14**, in each of which a variable is assigned the value $c-1$ or c (which values are bounded above by n), take time and space *linear* in ν, for similar reasons to line 7.

We summarize this analysis in Table 2.2.

Having considered the line-by-line complexity of N', we turn now to the method's overall complexity. Note that, most crucially, the depth of recursion is limited: given initial input n and having identified least prime factor $p \geq 2$ of n, the method calls itself with new argument $\frac{n}{p} \leq \frac{n}{2}$. The kth recursive call, then, is made with input $\frac{n}{p_1 p_2 p_3 \dots p_k} \leq \frac{n}{2^k}$ for some primes $p_i \geq 2$; further, each input is a natural number, and the method terminates at line 5 when a recursive call is made with prime input—this necessarily happens before the $(\log_2 n)$th recursive call, at which point the call's input value would be at most $\frac{n}{2^{\log_2 n}} = 1$. Therefore,

$$\textit{there are made } \mathcal{O}(\nu) \textit{ recursive calls};$$

we consider now the time and space complexity of each such.

Consider the variable $d := r_{\text{high}} - r_{\text{low}}$. After initialization of r_{low} and r_{high} at lines 6 and 7, $d = n-3$. In each repetition of the repeat loop (lines 8 – 15), either $d = 0$ (in which case a recursive call is made) or one of r_{low} and r_{high} is updated, either (a) the former to $c = \left\lceil \frac{r_{\text{low}}+r_{\text{high}}}{2} \right\rceil$ or (b) the latter to $c-1$. Case (a) represents a decrease of d by $\frac{d}{2}$ if r_{low} and r_{high} are of the same parity, and by $\frac{d+1}{2} \geq \frac{d}{2}$ otherwise; and (b) represents a decrease of d by $\frac{d+2}{2} \geq \frac{d}{2}$ if

Line:	Time complexity:	Space complexity:
1.	$\mathcal{O}(\nu)$	$\mathcal{O}(\nu)$
2.	$\mathcal{O}(1)$	$\mathcal{O}(1)$
3.	$\mathcal{O}(1)$	$\mathcal{O}(1)$
4.	$T_N^*(\nu)$	$S_N^*(\nu)$
5.	$\mathcal{O}(\nu)$	$\mathcal{O}(\nu)$
6.	$\mathcal{O}(1)$	$\mathcal{O}(1)$
7.	$\mathcal{O}(\nu)$	$\mathcal{O}(\nu)$
8.	$\mathcal{O}(1)$	$\mathcal{O}(1)$
9.	$\mathcal{O}(\nu)$	$\mathcal{O}(1)$
10.	$T_{N'}^*(\nu) + \mathcal{O}(\nu)$	$S_{N'}^*(\nu) + \mathcal{O}(\nu)$
11.	$\mathcal{O}(\nu)$	$\mathcal{O}(\nu)$
12.	$T_N^*(\nu)$	$S_N^*(\nu)$
13.	$\mathcal{O}(\nu)$	$\mathcal{O}(\nu)$
14.	$\mathcal{O}(\nu)$	$\mathcal{O}(\nu)$
15.	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Table 2.2: A summary of the line-by-line time and space complexity of method N'.

r_{low} and r_{high} are of the same parity, and by $\frac{d+1}{2} \geq \frac{d}{2}$ otherwise.[10] In any case, the value of d after k repetitions of the loop is at most $\frac{1}{2^k}$ of its initial value, namely, of $n-3$; after $\log_2 n$ repetitions, then, $d \leq \frac{n-3}{2^{\log_2 n}} = \frac{n-3}{n} < 1$, and, since d (the difference of two natural numbers) is, throughout, an integer, d becomes 0 (and, hence, N' has found another prime factor and will perform a recursive call) after at most $\log_2 n \in \mathcal{O}(\nu)$ repetitions of the repeat loop. Each pass through the loop that does not perform a recursive call at line 10 follows the path '9, 11, 12, 13' or '9, 11, 12, 14'; each takes time $T_N^*(\nu) + \mathcal{O}(\nu)$ and space $S_N^*(\nu) + \mathcal{O}(\nu)$.

The requisite repetitions to find a prime factor (that is, to perform one recursive call), then, take time $\mathcal{O}\left(\nu T_N^*(\nu) + \nu^2\right)$ *and space* $\mathcal{O}\left(\nu S_N^*(\nu) + \nu^2\right)$.

Consequently, since there are $\mathcal{O}(\nu)$ factors to find, the overall time complexity of N' is in $\mathcal{O}\left(\nu^2 T_N^*(\nu) + \nu^3\right)$ and the overall space complexity of N' in $\mathcal{O}\left(\nu^2 S_N^*(\nu) + \nu^3\right)$ (this includes also the 'initialization path' '1, 2, 4, 6, 7' for composite n). □

What we have seen is that the two versions of the factorization problem are equivalent in the sense that efficient (i.e., polynomial-resource) solution of either leads to efficient solution of the other.

We discuss in Chap. 3 the concept of *resource* in the context of computational complexity, arguing in particular that, especially when considering unconventional computers, consideration should be made of unconventional resources

[10] We make this explicit. (a) If r_{low} and r_{high} are of the same parity, then d is updated from $r_{\text{high}} - r_{\text{low}}$ to $r_{\text{high}} - c = r_{\text{high}} - \frac{r_{\text{low}}+r_{\text{high}}}{2} = \frac{r_{\text{high}}-r_{\text{low}}}{2}$; else, d is updated from $r_{\text{high}} - r_{\text{low}}$ to $r_{\text{high}} - c = r_{\text{high}} - \frac{r_{\text{low}}+r_{\text{high}}+1}{2} = \frac{r_{\text{high}}-r_{\text{low}}-1}{2}$. (b) If r_{low} and r_{high} are of the same parity, then d is updated from $r_{\text{high}} - r_{\text{low}}$ to $c - 1 - r_{\text{low}} = \frac{r_{\text{low}}+r_{\text{high}}-2}{2} - r_{\text{low}} = \frac{r_{\text{high}}-r_{\text{low}}-2}{2}$; else, d is updated from $r_{\text{high}} - r_{\text{low}}$ to $c - 1 - r_{\text{low}} = \frac{r_{\text{low}}+r_{\text{high}}-1}{2} - r_{\text{low}} = \frac{r_{\text{high}}-r_{\text{low}}-1}{2}$.

(rather than of time and space exclusively). Of interest in relation to this is that our method of proof of the equivalence of the two factorization problems—whereby we exhibit an efficient method for solving one given the ability to solve efficiently the other—is sufficiently general to guarantee preservation not only of time- and space-efficiency, but also of general resource-efficiency; the key observation to this end is that the processes combined with method M so as to produce M', and with N so as to produce N', are implementable by Turing machine, which implementation incurs only time and space costs. We formalize this in the following chapter (see Sect. 3.7.1), after our having introduced the general notion of resource.

Complexity of Factorization.

We see above that factorization as a *decision problem* is in a certain sense as hard as (and no harder than) its *function-problem* incarnation; how hard, in absolute rather than relative terms, are these problems?

First, it is clear that factorization (let us consider its decision-problem form) is in NP. Given an instance (n, l) that yields the answer 'yes', we have that there exists a factor a of n such that $1 < a < l$. A polynomially verifiable certificate for this instance, then, is given by a itself: one can confirm (via Turing machine) in polynomial time and space that a divides n.

Secondly, the problem is in co-NP. We recall from [108] that primes have polynomially verifiable certificates for primality. Hence, a certificate for a 'no' instance (n, l) of FACTORIZATION (DECISION) is provided by the multiset of prime factors of n together with primality certificates for these factors: it can be confirmed using only polynomial time and space that the product of the multiset's members is n, that each member is prime (via its primality certificate) and that each member is at least l.

We see, then, that

$$\textit{factorization is in } \mathsf{NP} \cap \mathsf{co\text{-}NP}.$$

Beyond this, disappointingly little is known.[11] The problem certainly seems difficult, however: we note from [41] that the run-time required to execute the best, known, *Turing-machine-style* methods grows as an exponential function of the number of digits of the value to be factorized;[12] this exponential time complexity renders computationally infeasible the factorization (via Turing machine) of numbers beyond a certain size. So, whilst FACTORIZATION (DECISION) is not known—is, further, suspected not[13]—to be NP-hard, neither do we expect an imminent, efficient algorithm.

FACTORIZATION (DECISION) appears not to be in P, then, as is suggested by the myriad mathematician-millennia that have failed to furnish efficient solution. The apparently likely situation in which factorization falls strictly between P

[11] As an aside, what *is* known (and has been only comparatively recently: the proof came in 2004 [4]) is that *ascertaining primality* or otherwise (though not necessarily finding factors) requires only polynomial time, and therefore polynomial space (since a Turing machine's space usage is bounded above by its run-time—see Footnote 109 of Chap. 3).

[12] Though [41] is, at time of writing, approximately a decade old, this exponential run-time remains state-of-the-art.

[13] In particular, if the problem were NP-complete (or, for that matter, co-NP-complete), then we should have the unlikely equality NP = co-NP.

and NP-completeness is, furthermore, perfectly possible in the sense that we have the following: Theorem 14.1 of [103] gives that,

> "[i]f P $\neq$ NP, then there is a language in NP which is neither in P nor is it NP-complete"

—NP cannot fall into a dichotomy of P on the one hand and the NP-complete on the other.

Whilst factorization via *Turing machine* seems difficult at a fundamental, theoretical level, then, the best, known, *quantum-computing* solutions to the problem are technologically hampered during implementation. Notably, Shor's quantum algorithm [118][14], despite having run-time polynomial in the size of the input value, has yet to factorize in practice a number greater than 15.[15]

Much faith has been placed in the difficulty of factorization (regardless of the computational model employed to tackle it). As a telling example, the extremely widely used Rivest-Shamir-Adleman (RSA) cryptographic system, introduced[16] in [112], relies for its security upon this apparent difficulty. We defer details of the system (e.g., to [132]), but note here that, crucially, the ability efficiently to factorize implies the ability efficiently to decode RSA-encrypted data without knowledge of the private key. The problem of factorization, then, has evaded efficient solution despite not only its millennia as a famous and natural mathematical problem, but also its important and incentive role in securing a vast number of electronic communications.

Aside. As a contextual aside, note that *number theory*, the mathematical field containing as a (comparatively tiny, distinctly proper) subset the study of factorization, captures much that is difficult about computation. Specifically, arbitrary Turing-machine computations can be encoded in diophantine equations in such a way as to imply that the (undecidable) Halting problem (see Sect. 1.2.1 *Halting Problem*) is no more difficult than the problem of deciding whether a given diophantine equation has a solution (see [48] for details). As [48] puts it,

> "this [reduction from Hilbert's Tenth problem to the Halting problem] *proves* that number theory is hard".

See also Sect. 5.3.

[14] We note as an aside that essentially the only use made by Shor's algorithm of non-classical (specifically, quantum) phenomena is in identifying the period of a certain periodic function (f, say); all else can, without impacting the system's computational complexity, be undertaken by a Turing-machine 'harness' supporting this non-classical process—cf. Remark 16. Further, we recall from [124] that slime-mould organisms can retain the period of a regular stimulus, such that provision of a later (possibly out-of-phase) stimulus (s, say) causes a response (r) exactly one period later. One naturally wonders, then, whether the slime-mould experiment can be modified such that the 'training stage'—wherein the period is imparted to the mould—is carried out by way of supplying not regular stimuli but rather some *encoding of function* f; this would allow the period of f to be found by later supplying stimulus s and timing the delay until response r. Since period-finding is the only non-classical task performed during Shor's algorithm, this would lead to a slime-mould rather than quantum implementation of Shor's factorization method. We defer to future work the exact details, and the complexity analysis, of the system.

[15] As an indication of the technological challenge faced here, we recall from [86] that, whilst digital computers can factorize numbers of 200 digits, a similar feat in the quantum-computing realm would require the successful implementation of *teraqubits*! (Recall that a *quantum bit* (or *qubit*) is the quantum analogue of a classical bit; see e.g., [99] for a formal definition.)

[16] In fact, only *publicly* was the system introduced in 1978 [112]; an equivalent form was discovered by Clifford Cocks at GCHQ in 1973.

Given the apparent difficulty of factorizing, especially within Turing and quantum models of computation, we naturally ask whether other models[17] (we here consider, specifically, analogue computers) offer a more efficient solution to the problem.[18] It is this question that motivates the analogue systems of Sects. 2.2 and 2.3.

2.2 Original Analogue Factorization System

We present an analogue system that factorizes natural numbers. Given input n, output from the system takes the form of all pairs of (not necessarily prime) natural numbers with product n, given which it is time- and space-efficient to find, via Turing machine, the prime factors of n, and, hence, solve FACTORIZATION (FUNCTION)[19]; thus, the problem solved by the system—specifically, finding factor pairs—is exactly as difficult as the two above-described formulations (function and decision) of the factorization problem.

The system is the subject of a United States patent [29], held by IBM and with sole inventor Ed Blakey; it is defined and discussed in [19], and presented (as in the current dissertation) as a motivating example in [18, 24]. Much of Sect. 2.2 is based on these cited works.

(The 'original' of this section's title is to distinguish the system from the subsequent, 'improved' version discussed in Sect. 2.3.)

2.2.1 Description

We describe now the original analogue factorization system. A clearly desirable property is that the system perform more efficiently than its digital-computer counterparts;[20] this is not possible by simply re-implementing an existing, Turing-machine-style algorithm's approach as an unconventional (e.g., analogue) computer. For example, a brute-force trial of each natural number in turn until a factor is found is, when performed by an analogue computer, still going to be subject to the exponential time complexity from which a corresponding digital computer would suffer; neither, we suggest, can such concerns be bypassed via relativistic computation or similar—see Sect. 3.6.1.

We note more generally that *efficient* use of unconventional-computing techniques is more likely to be a consequence of exploitation of 'what the computing

[17] We recall from [135] that, whilst *quantum* physics seemingly offers super-Turing computational efficiency, the availability or otherwise of such in *classical* physics is also an important issue.

[18] To those who like to read a whodunit's last page first, we say this: the analogue paradigm seems not in fact to facilitate efficient factorization (more's the pity), but the inefficiencies encountered do at least warrant study and suggest improvements upon the way in which complexity analyses are undertaken. We discuss this in due course.

[19] Explicitly, given all factors—prime or otherwise—of n, one can with only polynomial overhead identify the least, non-trivial such (p, say), which is necessarily prime; the analogue method can then be employed recursively with input $\frac{n}{p}$. The resultant system solves FACTORIZATION (FUNCTION); it has polynomial time/space complexity if and only if the analogue subsystem does (cf. line 10 of N' in the proof of Prop. 2).

[20] We see below that this is in some respects (specifically in terms of time complexity) achieved, though other concerns render the system unsuitable for practical use. That these concerns are indeed 'other'—that they do not form part of a standard complexity analysis—is our indicator that unconventional computers warrant unconventional complexity analysis, and our motivation for the complexity framework presented in the current project.

paradigm naturally wants to compute' than of a contrived attempt to implement a Turing-machine-style solution in the paradigm.[21] If, for instance, a ball rolling down a constant incline naturally computes squares (in that the distance rolled is proportional—with a discoverable constant of proportionality—to the square of the time elapsed), then it appears counterproductive to implement a squaring device by, say, having the ball roll along a network of channels isomorphic to the network of logic gates in a Turing-machine-style squarer; this latter approach seems not only counterproductive and destined not to improve upon the complexity of Turing-machine solutions, but also unilluminating with regard to both the squaring problem itself and the strengths of the unconventional (in this case, rolling-ball) paradigm.

Consequently, we should rather implement our analogue system in such a way that the relevant, naturally occurring phenomena are exploited than implement a 'digital-computer-style' algorithm via (incidentally and unadvantageously) analogue instantiations of logic gates, etc. This is made possible by the observation that

the factorization problem has a geometric formulation,

which we may implement comparatively directly as an analogue computer.

Geometric Formulation.

In this section, we reformulate as a geometric problem the numeric problem of factorization. This is possible because factorization of a natural number n may be restated as the search for integer solutions x and y to the equation $y = \frac{n}{x}$.

Proposition 3. The task of finding factors of a given natural number n is equivalent[22] to that of finding points that lie both in the integer grid $\mathbb{Z}^2$ and on the curve $y = \frac{n}{x}$.

Proof. A point $(a, b) \in \mathbb{R}^2$ is on the curve $y = \frac{n}{x}$ if and only if $n = ab$; the point is in the grid $\mathbb{Z}^2$ if and only if $a, b \in \mathbb{Z}$. Hence, (a, b) is both on the curve and in the grid if and only if a and b offer a factorization, into two integers, of n. □

See Fig. 2.1 for an example (specifically, $n = 6$) of a curve $y = \frac{n}{x}$ and its intersection with the integer grid.

Remark 1. The factorization of n corresponding to a point in the grid and on the curve is not necessarily—in fact, is rarely—a full decomposition of n into primes (it may even be no more informative than to demonstrate that $n = 1.n$ or that $n = n.1$). However, each prime factor p of n has a corresponding point $\left(p, \frac{n}{p}\right)$ in the grid and on the curve: all prime factors are represented by at fewest one such point each.

Remark 2. Since, by hypothesis, n is positive (we suppose, in particular, that $n \in \mathbb{N}$), the curve $y = \frac{n}{x}$ exists only in quadrants $x, y \geq 0$ and $x, y \leq 0$; further, since only positive factors of n (notably, of interest amongst these are the prime factors) are sought, only the former quadrant need be considered.

[21] We recall from [122] that Susan Stepney shares this view.

[22] The equivalence here is in the sense of computability rather than complexity: if one is *able* to find such points, then one is *able* to find factors (and, less crucially for our purposes, vice versa); we say nothing yet about these processes' efficiency.

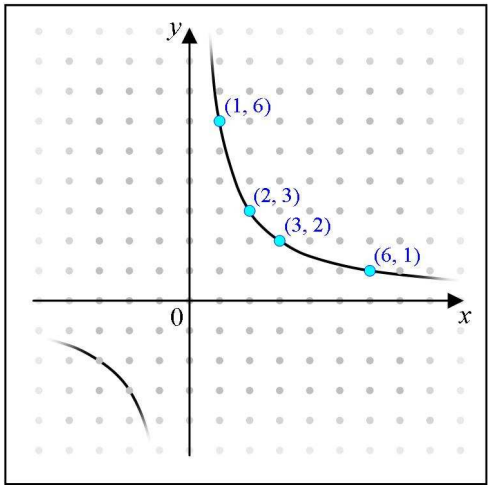

Figure 2.1: The curve $y = \frac{n}{x}$, with natural-number points thereon highlighted. In this example, $n = 6$.

Similarly, by the symmetry of the curve and of the integer grid—specifically, because each is symmetric about the line $y = x$—only one octant within this quadrant need be considered (since (a, b) is both on the curve and in the grid if and only if (b, a) is, and both points correspond, due to commutativity of multiplication, to the same partial factorization $n = ab$). Accordingly, we consider only the octant $0 \leq x \leq y$.

Proposition 4. The curve $y = \frac{n}{x}$, $z = 0$ (which lies in $\mathbb{R}^3$) can be expressed as the intersection of the (x, y)-plane and the cone[23] that consists of those lines that both pass through the point $\left(0, 0, \sqrt{2n}\right)$ and make an angle of $\frac{\pi}{4}$ of a radian with the line $y = x$, $z = \sqrt{2n}$.

Proof. Let C be the cone consisting of those lines that both pass through the point $P := \left(0, 0, \sqrt{2n}\right)$ (the vertex of the cone) and make an angle of $\frac{\pi}{4}$ of a radian with the line $y = x$, $z = \sqrt{2n}$ (call this line L).

Let $A := (a, b, 0) \in \mathbb{R}^3$ be an arbitrary point both on the cone C and in the (x, y)-plane; let B be the point $\left(x_A, x_A, \sqrt{2n}\right)$—hence, B is on L—, where $x_A = \frac{\left(a^2+b^2+2n\right)^{\frac{1}{2}}}{2}$.

We show first that B is equidistant from A and P.

Note that

$$\begin{aligned} |AP| &= \left(a^2 + b^2 + 2n\right)^{\frac{1}{2}} \\ &= 2x_A \\ &= \sqrt{2}\left(x_A^2 + x_A^2 + 0\right)^{\frac{1}{2}} \\ &= \sqrt{2}\,|BP| \quad . \end{aligned} \tag{2.5}$$

[23] That $y = \frac{n}{x}$, $z = 0$ is the intersection of the (x, y)-plane and *some* cone comes as no surprise: $y = \frac{n}{x}$ is a hyperbola, and, a fortiori, a conic section, which fact motivates the geometric formulation presented here.

Note further that, since A is on C and B on L, the definition of C gives that

$$\angle APB = \frac{\pi}{4} \ , \tag{2.6}$$

where $\triangle XYZ$ denotes the triangle with vertices at X, Y and Z, and $\angle XYZ$ denotes the angle interior to $\triangle XYZ$ at vertex Y (hence, $0 \leq \angle XYZ \leq \pi$).

Applying the cosine rule[24] to $\angle APB$ in $\triangle ABP$, then,

$$\begin{aligned} |AB|^2 &= |AP|^2 + |BP|^2 - 2\,|AP|\,|BP| \cos \angle APB \\ &\overset{(2.6)}{=} |AP|^2 + |BP|^2 - 2\,|AP|\,|BP| \cos \frac{\pi}{4} \\ &= |AP|^2 + |BP|^2 - 2\,|AP|\,|BP| \frac{1}{\sqrt{2}} \\ &\overset{(2.5)}{=} 2\,|BP|^2 + |BP|^2 - 2\sqrt{2}\,|BP|^2 \frac{1}{\sqrt{2}} \\ &= |BP|^2 \ . \end{aligned}$$

By non-negativity of $|AB|$ and $|BP|$, then,

$$|AB| = |BP| \ , \tag{2.7}$$

as claimed.

Hence,

$$\begin{aligned} \frac{\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}}}{\sqrt{2}} &= \frac{|AP|}{\sqrt{2}} \\ &\overset{(2.5)}{=} |BP| \\ &\overset{(2.7)}{=} |AB| \\ &= \left((a - x_A)^2 + (b - x_A)^2 + 2n\right)^{\frac{1}{2}} \\ &= \left(a^2 + b^2 - 2\,(a+b)\,x_A + 2x_A^2 + 2n\right)^{\frac{1}{2}} \ . \end{aligned}$$

Multiplying each side by $\sqrt{2}$,

$$\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}} = \left(2\left(a^2 + b^2 - 2\,(a+b)\,x_A + 2x_A^2 + 2n\right)\right)^{\frac{1}{2}} \ .$$

Squaring,

$$a^2 + b^2 + 2n = 2\left(a^2 + b^2 - 2\,(a+b)\,x_A + 2x_A^2 + 2n\right) \ .$$

Subtracting $a^2 + b^2 + 2n$ and recalling that $x_A = \frac{\left(a^2+b^2+2n\right)^{\frac{1}{2}}}{2}$,

$$\begin{aligned} 0 &= a^2 + b^2 - 4\,(a+b)\,x_A + 4x_A^2 + 2n \\ &= a^2 + b^2 - 2\,(a+b)\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}} + \left(a^2 + b^2 + 2n\right) + 2n \\ &= 2a^2 + 2b^2 + 4n - 2\,(a+b)\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}} \ . \end{aligned}$$

[24] Recall the cosine rule for $\angle XYZ$ in $\triangle XYZ$:

$$|XZ|^2 = |XY|^2 + |YZ|^2 - 2\,|XY|\,|YZ| \cos \angle XYZ \ .$$

Dividing by 2 and rearranging,

$$a^2 + b^2 + 2n = (a + b)\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}} \ .$$

Dividing by $\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}}$ (which is valid since a^2 and b^2 are non-negative and n is positive),

$$\left(a^2 + b^2 + 2n\right)^{\frac{1}{2}} = a + b \ .$$

Squaring,

$$\begin{aligned} a^2 + b^2 + 2n &= (a + b)^2 \\ &= a^2 + b^2 + 2ab \ . \end{aligned}$$

Hence, $ab = n$, and so

$A = (a, b, 0)$, an arbitrary point both on the cone C and in the (x, y)-plane, is on the curve $y = \frac{n}{x}$, $z = 0$.

Conversely, let $D := \left(d, \frac{n}{d}, 0\right) \in \mathbb{R}^3$ be an arbitrary point on the curve $y = \frac{n}{x}$, $z = 0$. Let E be the point $\left(x_D, x_D, \sqrt{2n}\right)$—$E$ is on L, hence—, where $x_D = \frac{d^2+n}{2d}$ (valid since d, the denominator of the y-coordinate of D, is non-zero). Then

$$\begin{aligned} |DE| &= \left(\left(d - \frac{d^2 + n}{2d}\right)^2 + \left(\frac{n}{d} - \frac{d^2 + n}{2d}\right)^2 + 2n\right)^{\frac{1}{2}} \\ &= \left(\left(\frac{d^2 - n}{2d}\right)^2 + \left(\frac{n - d^2}{2d}\right)^2 + 2n\right)^{\frac{1}{2}} \\ &= \left(2\left(\left(\frac{d^2 - n}{2d}\right)^2 + n\right)\right)^{\frac{1}{2}} \\ &= \left(2\left(\frac{d^4 - 2nd^2 + n^2}{4d^2} + \frac{4nd^2}{4d^2}\right)\right)^{\frac{1}{2}} \\ &= \left(2\left(\frac{d^4 + 2nd^2 + n^2}{4d^2}\right)\right)^{\frac{1}{2}} \\ &= \left(2\left(\frac{d^2 + n}{2d}\right)^2\right)^{\frac{1}{2}} \\ &= \frac{d^2 + n}{\sqrt{2}d} \ , \end{aligned}$$

$$\begin{aligned} |EP| &= \left(x_D^2 + x_D^2 + 0\right)^{\frac{1}{2}} \\ &= \sqrt{2}x_D \\ &= \frac{d^2 + n}{\sqrt{2}d} \ , \end{aligned}$$

and

$$\begin{aligned} |DP| &= \left(d^2 + \left(\frac{n}{d}\right)^2 + 2n\right)^{\frac{1}{2}} \\ &= \left(\frac{d^4 + n^2 + 2nd^2}{d^2}\right)^{\frac{1}{2}} \\ &= \frac{d^2 + n}{d} \ . \end{aligned}$$

Hence,

$$|DE| = |EP| = \frac{1}{\sqrt{2}} |DP| \ , \tag{2.8}$$

and so the cosine rule applied to $\angle DPE$ in $\triangle DEP$ gives that

$$\begin{aligned} \angle DPE &= \cos^{-1}\left(\frac{|DP|^2 + |EP|^2 - |DE|^2}{2\,|DP|\,|EP|}\right) \\ &\overset{(2.8)}{=} \cos^{-1}\left(\frac{|DP|^2 + \frac{1}{2}|DP|^2 - \frac{1}{2}|DP|^2}{2\,|DP|\,\frac{1}{\sqrt{2}}|DP|}\right) \\ &= \cos^{-1}\frac{1}{\sqrt{2}} \\ &= \frac{\pi}{4} \end{aligned}$$

(where the final equality holds since $0 \leq \angle DPE \leq \pi$—recall that the '$\angle$' notation denotes angles *interior* to triangles); $\angle DPE$—and, hence, the angle between DP and L, since E and P are on L—is $\frac{\pi}{4}$. Thus

D, an arbitrary point on the curve $y = \frac{n}{x}$, $z = 0$, is both on the cone C and in the (x, y)-plane,

as required. □

The physical implementation, which we now discuss, of the factorization method exploits the facts that factorization can be reformulated as the search for integer points on the curve $y = \frac{n}{x}$ (Prop. 3) and that this curve can be expressed as the intersection of a cone and a plane (Prop. 4).

Physical Implementation.

The Integer Grid.

Definition 2 (in which we implement the integer grid)**.** Let n be the natural number to be factorized; assume that n is odd—see Remark 3. Let ϵ be a small, positive, fixed real number $(0 < \epsilon \ll 1)$.[25]

1. Let M_1 be a parabolic mirror, reflective on its concave side, occupying the curve $\left\{ \left(x, -\frac{1}{2(1+\epsilon)}x^2 + x + (1+\epsilon), 0\right) \in \mathbb{R}^3 \;\middle|\; 0 \leq x \leq 1 \right\}$.

[25] In fact, we stipulate that $\epsilon \leq \frac{\sqrt{3}-1}{2} = 0.366\ldots$ for our convenience below.

2. Let M_2 be a plane mirror, reflective on its $x < y$ side, occupying the line segment $\{ (x, x, 0) \in \mathbb{R}^3 \mid 0 \leq x \leq 1 \}$.

3. Let M_3 be a plane mirror, reflective on its $x > 0$ side, occupying the line segment $\{ (0, y, 0) \in \mathbb{R}^3 \mid 0 \leq y \leq 1 \}$.

4. Let S be a source at $(1 + \epsilon, 1 + \epsilon, 0)$[Footnote 26] of sinusoidal, transverse-wave radiation with wavelength $\lambda := \frac{2}{n}$; suppose that S is shielded such that its radiation stays close to the plane $z = 0$.

5. Let B be a black body that absorbs radiation arriving from S and that occupies the region $\{ (x, y, 0) \in \mathbb{R}^3 \mid 1 \leq x \leq y < 1 + \epsilon \}$.

See Fig. 2.2.

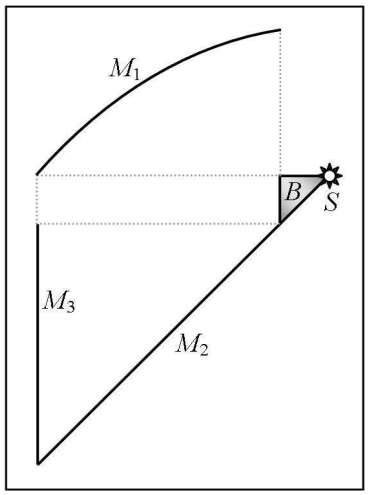

Figure 2.2: The layout of the apparatus that implements the integer grid (see Definition 2). Note that the value of ϵ has for clarity been exaggerated; it is shown here (and in relevant subsequent figures) as being approximately $\frac{1}{5}$, much larger than its actual value.

Remark 3. We assume that the number n to be factorized is odd.[27] This is because, for convenience, we implement the reduced grid $\{ (x, y) \mid x, y, \frac{x+y}{2} \in \mathbb{Z} \}$ (that is, pairs (x, y) of integers where the parity of x is that of y) instead of the full grid $\mathbb{Z}^2 = \{ (x, y) \mid x, y \in \mathbb{Z} \}$ mentioned in Sect. 2.2.1 *Geometric Formulation.* Any factorization of n (which is odd), then, into natural numbers x and y will be such that x and y are both odd, so this reduced grid suffices.

Further, consideration need be made only of that part of the reduced grid with $0 \leq x \leq y \leq n$ (since no factor of n is greater than n, and by Remark 2); only this part of the grid is implemented (see also Remark 7).

[26] S lies, therefore, at the focus of the parabola of which M_1 is part.

[27] Should a factorization of an even number be required, it is Turing-computationally trivial iteratively to divide by 2 until an odd number—which can be factorized as described here—is obtained. Since both testing for parity and division by 2 are polynomially achievable, and since the exponent of 2 in the prime factorization of n cannot exceed $\log_2 n$, the problems of natural-number factorization and odd-number factorization are equally difficult in the sense of Sect. 2.1.1 *The Problems' Equivalence.*

Remark 4. Since the wavelength $\frac{2}{n}$ of radiation from S depends upon n, its being set forms part of the computation's input process. More generally, a physical computing system will have manipulable parameters, which the user sets to values that encode his intended input value—recall Sect. 1.2.1 *Unconventional Computation*.

Proposition 5. Radiation incident on M_1 from S is reflected by M_1 as a beam of waves parallel to the y-axis, in the range $0 \leq x \leq 1$ (which is entirely spanned by such waves), and travelling in the direction of decreasing y.

Proof. (Readers for whom it suffices to note that S sits at the focus of the parabola containing M_1—recall Footnote 26—, that this parabola is symmetric about a line parallel to the y-axis—namely, $x = 1 + \epsilon$, $z = 0$—, and that the projection of M_1 onto the x-axis is the interval $[0, 1]$ may skip this proof.)

Since the region $\left\{ (x, y, 0) \in \mathbb{R}^3 \;\middle|\; y < -\frac{1}{2(1+\epsilon)}x^2 + x + (1+\epsilon) \right\}$ under the parabola containing M_1 is convex (this region is shown shaded in Fig. 2.3), since S lies in this region, and since no point of B lies on a line between S and any point on M_1 (because the line $y = 1 + \epsilon$, $z = 0$, which passes through S, separates B and M_1), there is radiation from S incident on each point of M_1.

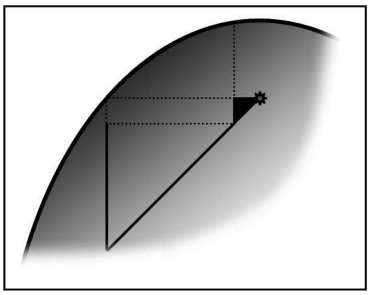

Figure 2.3: The region $\left\{ (x, y, 0) \in \mathbb{R}^3 \;\middle|\; y < -\frac{1}{2(1+\epsilon)}x^2 + x + (1+\epsilon) \right\}$ (shaded) under the parabola containing M_1. The layout of Fig. 2.2 is reproduced for context.

Consider the radiation incident on an arbitrary point A of M_1; say $A = \left(a, -\frac{1}{2(1+\epsilon)}a^2 + a + (1+\epsilon), 0\right)$ with $0 \leq a \leq 1$. See Fig. 2.4 for the layout of A and other constructions (t_A, n_A, u_A, T and S') used in this proof.

The gradient of the curve $y = -\frac{1}{2(1+\epsilon)}x^2 + x + (1+\epsilon)$ is given by $y' = -\frac{1}{1+\epsilon}x + 1$, which at A is $1 - \frac{a}{1+\epsilon}$, so the tangent t_A at A to the curve has equation $y = \left(1 - \frac{a}{1+\epsilon}\right)x + \frac{a^2}{2(1+\epsilon)} + 1 + \epsilon$, $z = 0$ and the normal n_A at A to the curve has equation $y = \left(\frac{1+\epsilon}{a-1-\epsilon}\right)x + a + 1 + \epsilon + \frac{a(1+\epsilon)}{1+\epsilon-a} - \frac{a^2}{2(1+\epsilon)}$, $z = 0$. The radiation from S incident on A is reflected along the line passing through the reflections in n_A of S and of A; this is necessary for the radiation's angles of incidence on and reflection in M_1 to be equal.

Let u_A be the line parallel to t_A and passing through S; this has equation $y = \left(1 - \frac{a}{1+\epsilon}\right)x + a$, $z = 0$. Let T be the point on n_A and u_A. By construction (and, specifically, since n_A and u_A are perpendicular and since u_A passes through S), T is the midpoint between S and the reflection in n_A of S (call this reflection

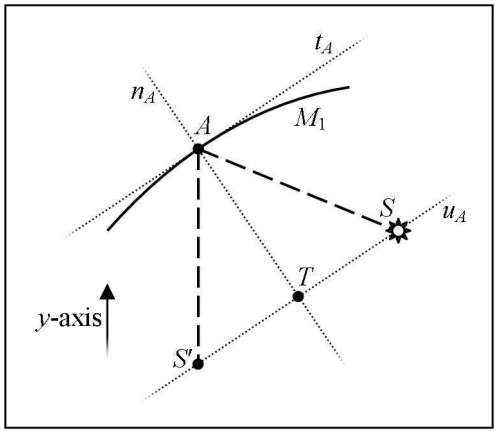

Figure 2.4: The layout of constructions used in the proof of Prop. 5.

S'). In particular, letting P_x denote the x-coordinate of a point P, T_x is the mean of S_x and S'_x; therefore,

$$S'_x = 2T_x - S_x \ . \tag{2.9}$$

Recall from point 4 of Definition 2 that

$$S_x = 1 + \epsilon \ . \tag{2.10}$$

T_x is the value of x for which $y = \left(1 - \frac{a}{1+\epsilon}\right) x + a$, $z = 0$ (i.e. u_A) meets $y = \left(\frac{1+\epsilon}{a-1-\epsilon}\right) x + a + 1 + \epsilon + \frac{a(1+\epsilon)}{1+\epsilon-a} - \frac{a^2}{2(1+\epsilon)}$, $z = 0$ (i.e. n_A); that is, T_x is such that $\left(1 - \frac{a}{1+\epsilon}\right) T_x + a = \left(\frac{1+\epsilon}{a-1-\epsilon}\right) T_x + a + 1 + \epsilon + \frac{a(1+\epsilon)}{1+\epsilon-a} - \frac{a^2}{2(1+\epsilon)}$. Rearranging,

we have that

$$\begin{aligned}
T_x &= \left(a+1+\epsilon+\frac{a\,(1+\epsilon)}{1+\epsilon-a}-\frac{a^2}{2\,(1+\epsilon)}-a\right) \\
&\quad \div\left(1-\frac{a}{1+\epsilon}-\frac{1+\epsilon}{a-1-\epsilon}\right) \\
&= \left(1+\epsilon+\frac{a\,(1+\epsilon)}{1+\epsilon-a}-\frac{a^2}{2\,(1+\epsilon)}\right)\div\left(1-\frac{a}{1+\epsilon}+\frac{1+\epsilon}{1+\epsilon-a}\right) \\
&= \frac{2\,(1+\epsilon-a)\,(1+\epsilon)^2+2a\,(1+\epsilon)^2-a^2\,(1+\epsilon-a)}{2\,(1+\epsilon-a)\,(1+\epsilon)} \\
&\quad \times\frac{(1+\epsilon-a)\,(1+\epsilon)}{(1+\epsilon-a)\,(1+\epsilon)-a\,(1+\epsilon-a)+(1+\epsilon)^2} \\
&= \frac{2\,(1+\epsilon-a)\,(1+\epsilon)^2+2a\,(1+\epsilon)^2-a^2\,(1+\epsilon-a)}{2\,(1+\epsilon-a)\,(1+\epsilon)-2a\,(1+\epsilon-a)+2\,(1+\epsilon)^2} \\
&= \big(2+4\epsilon+2\epsilon^2+2\epsilon+4\epsilon^2+2\epsilon^3-2a-4\epsilon a \\
&\quad -2\epsilon^2 a+2a+4\epsilon a+2\epsilon^2 a-a^2-\epsilon a^2+a^3\big) \\
&\quad \div\big(2+2\epsilon+2\epsilon+2\epsilon^2-2a-2\epsilon a-2a-2\epsilon a+2a^2+2+4\epsilon+2\epsilon^2\big) \\
&= \frac{2+6\epsilon+6\epsilon^2+2\epsilon^3-a^2-\epsilon a^2+a^3}{2\,(2+4\epsilon+2\epsilon^2-2a-2\epsilon a+a^2)}\ .
\end{aligned} \tag{2.11}$$

Hence,

$$\begin{aligned}
S'_x &\overset{(2.9)}{=} 2T_x-S_x \\
&\overset{(2.11,\,2.10)}{=} \frac{2+6\epsilon+6\epsilon^2+2\epsilon^3-a^2-\epsilon a^2+a^3}{2+4\epsilon+2\epsilon^2-2a-2\epsilon a+a^2}-1-\epsilon \\
&= \big(2+6\epsilon+6\epsilon^2+2\epsilon^3-a^2-\epsilon a^2+a^3-2-4\epsilon-2\epsilon^2+2a \\
&\quad +2\epsilon a-a^2-2\epsilon-4\epsilon^2-2\epsilon^3+2\epsilon a+2\epsilon^2 a-\epsilon a^2\big) \\
&\quad \div\big(2+4\epsilon+2\epsilon^2-2a-2\epsilon a+a^2\big) \\
&= \frac{-2a^2-2\epsilon a^2+a^3+2a+4\epsilon a+2\epsilon^2 a}{2+4\epsilon+2\epsilon^2-2a-2\epsilon a+a^2} \\
&= \frac{a\,\big(2+4\epsilon+2\epsilon^2-2a-2\epsilon a+a^2\big)}{2+4\epsilon+2\epsilon^2-2a-2\epsilon a+a^2} \\
&= a\ .
\end{aligned}$$

So $S'_x = A_x = a$, and the line of the reflected radiation passes through points A and S' (which are distinct, being separated by the same distance as are A and S). Hence, the reflected radiation passes along the line $x = a$, $z = 0$ (parallel to the y-axis), from A towards S'—i.e., with y decreasing—, as required.

Further, since each point on M_1 has incident radiation (recall the second paragraph of this proof), the range $0 \le x \le 1$ is entirely spanned by reflected waves. □

Remark 5. Radiation from S not incident on M_1 is not of interest here; it is either absorbed by B or completely leaves the apparatus.

Proposition 6. The beam described in Prop. 5 is reflected by M_2 to form a beam parallel to the x-axis, in the range $0 \leq y \leq 1$ (which is entirely spanned by the reflected beam), and travelling in the direction of decreasing x.

Proof. By Prop. 5, the part of the incoming beam incident on an arbitrary point $A := (a, a, 0)$ (with $0 \leq a \leq 1$) of M_2 travels to A along the line $x = a$, $z = 0$, with y decreasing. This is reflected by M_2, which sits at an angle of $\frac{\pi}{4}$ to the x- and y-axes, along the line $y = a$, $z = 0$, with x decreasing.

Further, the reflected beam spans the range $0 \leq y \leq 1$ since, by Prop. 5, the incoming beam spans $0 \leq x \leq 1$. □

Proposition 7. Radiation incident on M_3 from S (via M_1 and M_2) is reflected by M_3 back along itself, producing a standing wave.

Proof. By Prop. 6, radiation reaches a point $A := (0, a, 0)$ (with $0 \leq a \leq 1$) of M_3 by travelling along the line $y = a$, $z = 0$, with x decreasing. Since this line is parallel to the x-axis and M_3 to the y-axis (and since both are within the (x, y)-plane), the incident ray is normal to the mirror and is reflected along itself. The nature of the standing wave thus produced is described in Prop. 8. □

Remark 6 (in which we summarize Props. 5, 6 and 7). A ray from S that is of interest (that falls, that is, on mirror M_1 rather than leaving the apparatus or being absorbed by B) meets M_1 at the point $\left(a, -\frac{1}{2(1+\epsilon)}a^2 + a + (1+\epsilon), 0\right)$ for some $0 \leq a \leq 1$ (conversely, each such a has a corresponding ray). It is then reflected by M_1 vertically down to $(a, a, 0)$, where M_2 reflects it horizontally across to $(0, a, 0)$. M_3 then reflects the ray back along itself via M_2 and M_1 to S, setting up a standing wave, which is described below. See Fig. 2.5 for the route of propagation of a sample ray.

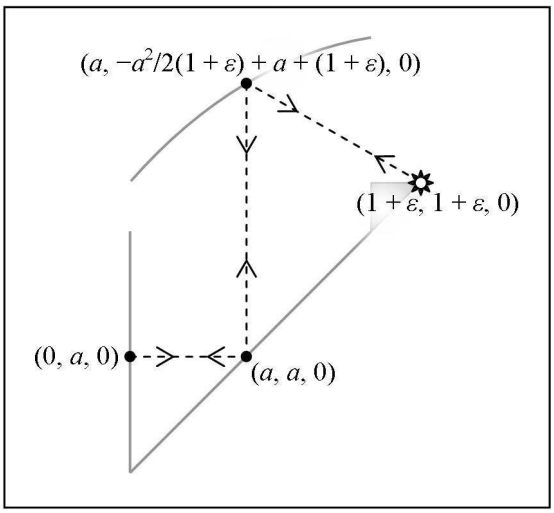

Figure 2.5: The path of a ray propagating from S via the three mirrors M_i back to S.

Proposition 8. In the triangular region $R := \{ (x, y, 0) \in \mathbb{R}^3 \mid 0 \leq x \leq y \leq 1 \}$, the interference pattern produced by the standing waves detailed above is such that a point $(a, b, 0)$ in R is at maximum amplitude (specifically, four times the amplitude of the original radiation from S) if and only if na and nb are integers of the same parity.

(By way of example, Fig. 2.6 shows these points of maximal wave activity within the region R in the case $n = 9$.)

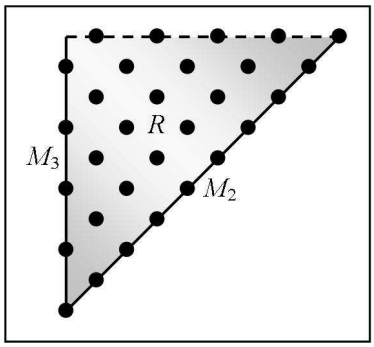

Figure 2.6: The region R, and the points—shown as dots—of maximal wave activity therein (in this example, $n = 9$).

Proof. (For brevity, we define the function $f\colon [0, 1] \to \mathbb{R}$ by

$$f\colon x \mapsto -\frac{x^2}{2\,(1+\epsilon)} + x + 1 + \epsilon \ .$$

Therefore, parabolic mirror M_1 occupies the curve $\{ (x, f(x), 0) \mid 0 \leq x \leq 1 \}$, and the Euclidean distance between the points $\left(x, -\frac{x^2}{2(1+\epsilon)} + x + 1 + \epsilon, 0\right)$ (on M_1) and $(x, 0, 0)$ is $f(x)$.)

By Props. 5, 6 and 7, we have that radiation within the region R propagates parallel to either the x- or the y-axis. Consequently, the interference pattern at a point $(a, b, 0)$ in R can be influenced by radiation arriving only from the four cardinal points—the four rays of interest, then, are

1. that—shown in yellow (■) in Fig. 2.7—from S via $(a, f(a), 0)$ on M_1 to $(a, b, 0)$;
2. that—shown in blue (■)—from S via $(a, f(a), 0)$ on M_1, $(a, a, 0)$ on M_2, $(0, a, 0)$ on M_3 and $(a, a, 0)$ on M_2 to $(a, b, 0)$;
3. that—shown in green (■)—from S via $(b, f(b), 0)$ on M_1 and $(b, b, 0)$ on M_2 to $(a, b, 0)$; and
4. that—shown in red (■)—from S via $(b, f(b), 0)$ on M_1, $(b, b, 0)$ on M_2 and $(0, b, 0)$ on M_3 to $(a, b, 0)$.

Since these rays are sinusoidal—recall point 4 of Definition 2—, we may model the amplitude at the point $(a, b, 0)$ of each of the rays by

$$\alpha \sin\left(\frac{2\pi}{\lambda} d - t\right) \ ,$$

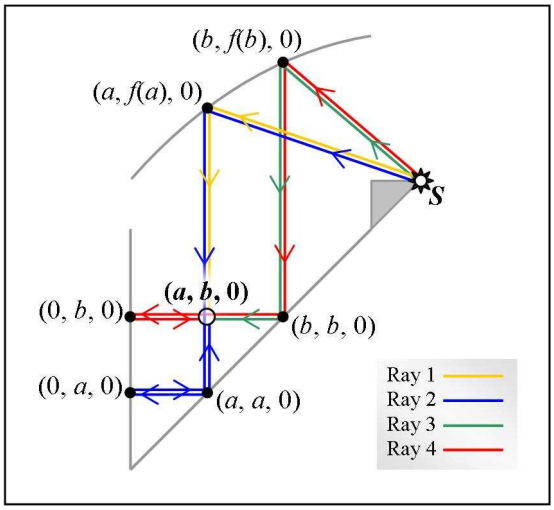

Figure 2.7: The four rays incident on point $(a, b, 0)$.

where

- α is the amplitude of the original radiation from S,
- λ its wavelength,
- d the total distance travelled by the ray from S via points on mirrors M_i to $(a, b, 0)$, and
- $t \geq 0$ a real-number model of time.

Writing, again for brevity, $g(x)$ for $\left((1+\epsilon-x)^2 + (1+\epsilon-f(x))^2\right)^{\frac{1}{2}} + f(x)$ (for $0 \leq x \leq 1$),[28] the respective values of d for the four rays are $d_1 := g(a) - b$, $d_2 := g(a) + b$, $d_3 := g(b) - a$ and $d_4 := g(b) + a$; this is apparent from Figs. 2.7 and 2.8.

Note that these expressions for d_i can be simplified since $g(x) = 2(1+\epsilon)$ for each $x \in [0, 1]$, as we demonstrate in Lemma 4 below. Explicitly, we have that

$$\begin{aligned} d_1 &= 2(1+\epsilon) - b \ , \\ d_2 &= 2(1+\epsilon) + b \ , \\ d_3 &= 2(1+\epsilon) - a \ \text{ and} \\ d_4 &= 2(1+\epsilon) + a \ . \end{aligned} \tag{2.12}$$

Recall the sum-to-product identity for sines: for $\theta, \phi \in \mathbb{R}$,

$$\sin\theta + \sin\phi = 2\sin\frac{\theta+\phi}{2}\cos\frac{\theta-\phi}{2} \ ; \tag{2.13}$$

recall also that

$$\cos\theta = \cos(-\theta) \ , \tag{2.14}$$

[28] Hence, $g(x)$ is the total of the Euclidean distances between S and $(x, f(x), 0)$ and between $(x, f(x), 0)$ and $(x, 0, 0)$. See Fig. 2.8.

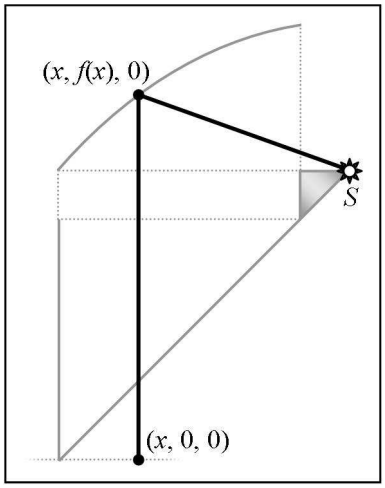

Figure 2.8: Two line segments of which the combined length is $g(x) = 2(1+\epsilon)$.

and that the wavelength of radiation from S is

$$\lambda = \frac{2}{n} \ . \tag{2.15}$$

Then the resultant amplitude $\sum_{i=1}^{4} \alpha \sin\left(\frac{2\pi}{\lambda} d_i - t\right)$ at $(a, b, 0)$ can be written as

$$\begin{aligned}
& \sum_{i=1}^{4} \alpha \sin\left(\frac{2\pi}{\lambda} d_i - t\right) \\
= \quad & \sum_{i=1}^{2} \alpha \sin\left(\frac{2\pi}{\lambda} d_i - t\right) + \sum_{j=3}^{4} \alpha \sin\left(\frac{2\pi}{\lambda} d_j - t\right) \\
\overset{(2.13)}{=} \quad & 2\alpha \sin\left(\frac{2\pi}{\lambda} \cdot \frac{d_1 + d_2}{2} - t\right) \cos\left(\frac{2\pi}{\lambda} \cdot \frac{d_1 - d_2}{2}\right) \\
& + 2\alpha \sin\left(\frac{2\pi}{\lambda} \cdot \frac{d_3 + d_4}{2} - t\right) \cos\left(\frac{2\pi}{\lambda} \cdot \frac{d_3 - d_4}{2}\right) \\
\overset{(2.12,\ 2.14)}{=} \quad & 2\alpha \left(\sin\left(\frac{2\pi}{\lambda} 2(1+\epsilon) - t\right) \cos\left(\frac{2\pi}{\lambda} b\right)\right. \\
& \left. + \sin\left(\frac{2\pi}{\lambda} 2(1+\epsilon) - t\right) \cos\left(\frac{2\pi}{\lambda} a\right)\right) \\
\overset{(2.15)}{=} \quad & 2\alpha \left(\sin\left(2n\pi(1+\epsilon) - t\right) \cos\left(n\pi b\right)\right. \\
& \left. + \sin\left(2n\pi(1+\epsilon) - t\right) \cos\left(n\pi a\right)\right) \ .
\end{aligned}$$

Note that, since the sine and cosine functions both return values in the interval $[-1, 1]$, this amplitude is in the interval $[-4\alpha, 4\alpha]$. The result to be proven is that the amplitude attains its maximum (within R) at point $(a, b, 0)$ if and only if na and nb are integers of the same parity; by the previous sentence, it is sufficient for maximality at $(a, b, 0)$ that the amplitude at $(a, b, 0)$ attain the value 4α (for some $t \geq 0$).

Let $(a, b, 0)$ be a point in R such that na and nb are integers of the same parity; suppose first that na and nb are even; let $t_0 = \pi\left(2n\left(1+\epsilon\right) - \frac{1}{2}\right)$. Then $n\pi a$ and $n\pi b$ are even multiples of π, and so $\cos\left(n\pi a\right) = \cos\left(n\pi b\right) = 1$. Further,

$$\begin{aligned}\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right) &= \sin\left(2n\pi\left(1+\epsilon\right) - \pi\left(2n\left(1+\epsilon\right) - \frac{1}{2}\right)\right) \\ &= \sin\left(\frac{\pi}{2}\right) \\ &= 1\ ,\end{aligned}$$

so

$$\begin{aligned}&2\alpha\left(\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right)\cos\left(n\pi b\right)\right. \\ &\left.+\ \sin\left(2n\pi\left(1+\epsilon\right) - t_0\right)\cos\left(n\pi a\right)\right) &= 2\alpha\left(1\times 1 + 1\times 1\right) \\ &&= 4\alpha\ ;\end{aligned}$$

that is, if na and nb are both even, then the amplitude at $(a, b, 0)$ is 4α. Suppose instead that na and nb are both odd, and let $t_0 = \pi\left(2n\left(1+\epsilon\right) + \frac{1}{2}\right)$. Then $n\pi a$ and $n\pi b$ are odd multiples of π, and so $\cos\left(n\pi a\right) = \cos\left(n\pi b\right) = -1$. Further,

$$\begin{aligned}\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right) &= \sin\left(2n\pi\left(1+\epsilon\right) - \pi\left(2n\left(1+\epsilon\right) + \frac{1}{2}\right)\right) \\ &= \sin\left(-\frac{\pi}{2}\right) \\ &= -1\ ,\end{aligned}$$

so

$$\begin{aligned}&2\alpha\left(\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right)\cos\left(n\pi b\right)\right. \\ &\left.+\ \sin\left(2n\pi\left(1+\epsilon\right) - t_0\right)\cos\left(n\pi a\right)\right) &= 2\alpha\left(\left(-1\right)^2 + \left(-1\right)^2\right) \\ &&= 4\alpha\ ;\end{aligned}$$

that is, if na and nb are both odd, then the amplitude at $(a, b, 0)$ is 4α. Whether both odd or both even, then,

if na and nb are integers of the same parity, then the amplitude at $(a, b, 0)$ is 4α.

Conversely, suppose that point $(a, b, 0)$ in R has amplitude 4α (say that t_0 satisfies $2\alpha\left(\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right)\cos\left(n\pi b\right) + \sin\left(2n\pi\left(1+\epsilon\right) - t_0\right)\cos\left(n\pi a\right)\right) = 4\alpha$); then

$$\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right) = \cos\left(n\pi b\right) \in \{\pm 1\} \tag{2.16}$$

and

$$\sin\left(2n\pi\left(1+\epsilon\right) - t_0\right) = \cos\left(n\pi a\right) \in \{\pm 1\}\ . \tag{2.17}$$

So, since $\cos\left(\pi nb\right)$ and $\cos\left(\pi na\right)$ are in $\{\pm 1\}$, na and nb are integers. Required is that na and nb have the same parity. Now

$$\begin{aligned}\cos\left(n\pi b\right) &\overset{(2.16)}{=} \sin\left(2n\pi\left(1+\epsilon\right) - t_0\right) \\ &\overset{(2.17)}{=} \cos\left(n\pi a\right) \\ &= \begin{cases} 1 & \text{if } na \text{ is even} \\ -1 & \text{if } na \text{ is odd}\ . \end{cases}\end{aligned}$$

Hence, nb has the same parity as na.

If $(a, b, 0) \in R$ displays maximal amplitude, then a and b are integers of the same parity,

as required. □

Lemma 4. If $x \in [0, 1]$, then $g(x) = 2(1 + \epsilon)$.

Proof. Recall that, for any $x \in [0, 1]$,

$$f(x) = -\frac{x^2}{2(1+\epsilon)} + x + 1 + \epsilon \tag{2.18}$$

and

$$g(x) = \left((1 + \epsilon - x)^2 + (1 + \epsilon - f(x))^2\right)^{\frac{1}{2}} + f(x) \quad . \tag{2.19}$$

By (2.18), $1 + \epsilon - f(x) = \frac{x^2}{2(1+\epsilon)} - x$, whence (subtracting x and multiplying by $2(1 + \epsilon)$)

$$2(1 + \epsilon)(1 + \epsilon - f(x) - x) = x^2 - 4x(1 + \epsilon) \quad . \tag{2.20}$$

Also, again by (2.18), $f(x)^2 = \frac{x^4}{4(1+\epsilon)^2} - \frac{x^3}{1+\epsilon} + 2x(1 + \epsilon) + (1 + \epsilon)^2$, whence (adding $2x^2 - 4x(1 + \epsilon)$)

$$2x^2 - 4x(1 + \epsilon) + f(x)^2 = \frac{x^4}{4(1+\epsilon)^2} - \frac{x^3}{1+\epsilon} + 2x^2 - 2x(1 + \epsilon) + (1 + \epsilon)^2 \quad . \tag{2.21}$$

Hence, for any $x \in [0, 1]$,

$$\begin{aligned}
& g(x) \\
\overset{(2.19)}{=} & \left[(1 + \epsilon - x)^2 + (1 + \epsilon - f(x))^2\right]^{\frac{1}{2}} + f(x) \\
= & \left[2(1 + \epsilon)^2 - 2(1 + \epsilon)(x + f(x)) + x^2 + f(x)^2\right]^{\frac{1}{2}} + f(x) \\
= & \left[2(1 + \epsilon)(1 + \epsilon - f(x) - x) + x^2 + f(x)^2\right]^{\frac{1}{2}} + f(x) \\
\overset{(2.20)}{=} & \left[2x^2 - 4x(1 + \epsilon) + f(x)^2\right]^{\frac{1}{2}} + f(x) \\
\overset{(2.21)}{=} & \left[\frac{1}{4}x^4(1 + \epsilon)^{-2} - x^3(1 + \epsilon)^{-1} + 2x^2 - 2x(1 + \epsilon) + (1 + \epsilon)^2\right]^{\frac{1}{2}} + f(x) \\
= & \left[\left(\frac{x^2}{2(1+\epsilon)} - x + 1 + \epsilon\right)^2\right]^{\frac{1}{2}} + f(x) \\
= & \frac{x^2}{2(1+\epsilon)} - x + 1 + \epsilon + f(x) \\
\overset{(2.18)}{=} & 2(1 + \epsilon) \quad ,
\end{aligned}$$

as claimed. □

Remark 7. The set $\left\{ \left(\frac{x}{n}, \frac{y}{n}, 0\right) \in R \;\middle|\; x, y, \frac{x+y}{2} \in \mathbb{Z} \right\}$ of high-amplitude points of the interference pattern in R (see Prop. 8) models in the obvious way the reduced grid $\left\{ (x, y) \;\middle|\; x, y, \frac{x+y}{2} \in \mathbb{Z} \wedge 0 \leq x \leq y \leq n \right\}$ described in Remark 3: a point $\left(\frac{a}{n}, \frac{b}{n}, 0\right)$ in the former models (a, b) in the latter. (In fact, the whole region R, of which continuum the high-amplitude points are a point-lattice subset, corresponds under the same transformation $\left(\left(\frac{x}{n}, \frac{y}{n}, 0\right) \mapsto (x, y)\right)$ to the region $\left\{ (x, y) \in \mathbb{R}^2 \;\middle|\; 0 \leq x \leq y \leq n \right\}$, of which the reduced grid is a subset.)

This change of scale, by a multiplicative factor of n, of the x- and y-axes is carried out in order that the dimensions and layout of the apparatus described be independent of the choice of n, in practice allowing use of the same apparatus for different values of n; cf. the modification, from Definition 4 to Definition 5, of the improved factorization system of Sect. 2.3.

The Cone.

Definition 3 (in which we implement the cone)**.**

1. Let P_n be a source at $\left(0, 0, \sqrt{\frac{2}{n}}\right)$ of radiation.
2. Let C_n be a sensor, capable of detecting radiation from P_n, along the curve

$$\left\{ (x, 2-x, z) \in \mathbb{R}^3 \;\middle|\; \begin{array}{ll} & 2(x-1)^2 + \left(z - \sqrt{\frac{2}{n}}\right)^2 = 2 \\ \wedge & z \leq \frac{1-n}{1+n}\sqrt{\frac{2}{n}} \\ \wedge & 2 - x \geq 1 \end{array} \right\} .$$

See Fig. 2.9.

Remark 8. The subscripts '$_n$' in Definition 3 reflect the fact that the spatial positions of P_n and C_n depend upon n; the positioning of these components, therefore, forms part of the input process for the computation, as we describe in points 1 and 2 of Sect. 2.2.3 *Using the System.*

Remark 9. On a note related to Remark 8, were we proposing that the factorizing system be implemented for real-life use[29], then one concern regarding C_n would be that not merely its position, but also its *structure*, depends upon n: having used the system to factorize some value, subsequently to factorize a different value would necessitate not only moving, but also modifying, C_n. However, the curve occupied by C_n is, regardless of n, an arc of a circle[30] of radius $\sqrt{2}$ within the plane $y = 2 - x$ and with its centre on the line $x = y = 1$; further, the presence of a sensor occupying the full circle (rather than just this arc) is not problematic for the functioning of the system. Therefore, we may implement C_n as a (full-) circular sensor (from which readings are considered

[29]We are certainly not suggesting that the system offers a practical method of factorizing large numbers; of interest here is the *reason for which it does not offer such*, and, in particular, that this reason should, but does not, constitute part of standard complexity theory.

[30]Specifically, the circle is $y = 2 - x$, $2(x-1)^2 + \left(z - \sqrt{\frac{2}{n}}\right)^2 = 2$, and the arc consists of those points on the circle such that $z \leq \frac{1-n}{1+n}\sqrt{\frac{2}{n}}$ and $x \leq 1$.

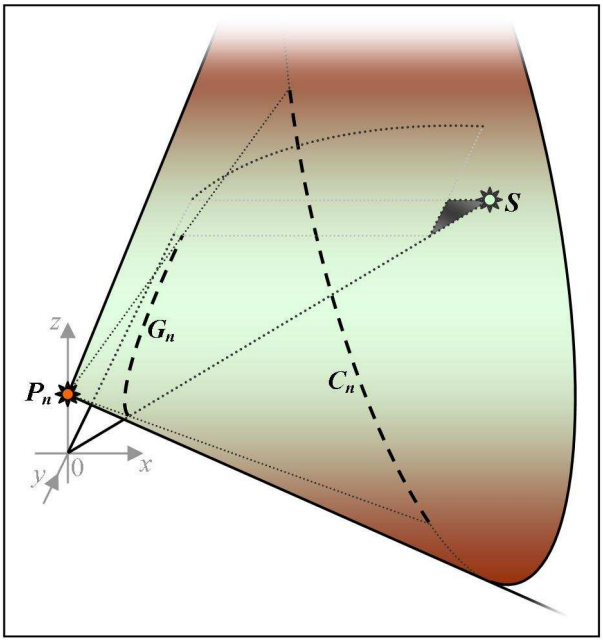

Figure 2.9: The layout of the apparatus that implements the cone (see Definition 3). Also shown are the apparatus of Fig. 2.2 (for context) and the curve G_n (see Prop. 9).

if they concern points on the arc and ignored otherwise), of which the position (specifically, the z-axis height), but not the structure, depends upon n.

In the present discussion—which, we reiterate, includes this analogue factorization system as motivation for new approaches to complexity theory rather than as a proposal for a practicably implementable system—, we define C_n as in Definition 3 rather than treating it as occupying the whole circle.

Proposition 9. The curve of C_n is the circular arc produced by projecting the curve $G_n := \left\{ (x, y, 0) \in R \;\middle|\; \frac{1}{xy} = n \right\}$ from P_n onto the plane $y = 2 - x$. Hence, radiation arriving from P_n at a point on C_n has passed through the plane $z = 0$ at a point $(x, y, 0)$ such that $\frac{1}{xy} = n$.

See Fig. 2.10 for the position within R of the curve G_n for various values of n.

Proof. Let $(a, b, 0)$ be an arbitrary point on G_n; then $\frac{1}{ab} = n$, so this point is $\left(a, \frac{1}{na}, 0\right)$.[31] The line that passes through both $\left(a, \frac{1}{na}, 0\right)$ and $P_n = \left(0, 0, \sqrt{\frac{2}{n}}\right)$ is given by $\left\{ \left(a, \frac{1}{na}, 0\right) + \gamma \left(a, \frac{1}{na}, -\sqrt{\frac{2}{n}}\right) \;\middle|\; \gamma \in \mathbb{R} \right\}$; this is equal to

$$\left\{ \left((\gamma + 1)\, a, \frac{\gamma + 1}{na}, -\gamma \sqrt{\frac{2}{n}} \right) \;\middle|\; \gamma \in \mathbb{R} \right\} \quad .$$

[31] That this is indeed the same point follows from the fact that neither a nor b is zero, which, in turn, is because $ab = \frac{1}{n} > 0$.

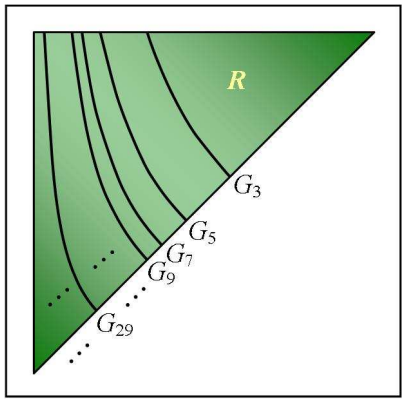

Figure 2.10: The position within R of G_n, for $n = 3$, 5, 7, 9 and 29.

This line meets the plane defined by $y = 2 - x$ when $\frac{\gamma+1}{na} = 2 - (\gamma + 1)\, a$ (i.e., when $\gamma + 1 = \frac{2na}{1+na^2}$); this happens at the point

$$A := \left(\frac{2na^2}{1+na^2}, \frac{2}{1+na^2}, \frac{1+na^2-2na}{1+na^2}\sqrt{\frac{2}{n}} \right) \ ,$$

which, we claim, lies on the curve of C_n. Recall that this curve consists of those points $(x, 2-x, z) \in \mathbb{R}^3$ such that

$$2\,(x-1)^2 + \left(z - \sqrt{\frac{2}{n}} \right)^2 = 2 \ , \tag{2.22}$$

$$z \leq \frac{1-n}{1+n}\sqrt{\frac{2}{n}} \quad \text{and} \tag{2.23}$$

$$2 - x \geq 1 \ . \tag{2.24}$$

First, A satisfies (2.22), since

$$\begin{aligned} 2\left(\frac{2na^2}{1+na^2} - 1\right)^2 & \\ + \left(\frac{1+na^2-2na}{1+na^2}\sqrt{\frac{2}{n}} - \sqrt{\frac{2}{n}}\right)^2 &= 2\left(\frac{2na^2-1-na^2}{1+na^2}\right)^2 \\ &\quad + \frac{2}{n}\left(\frac{1+na^2-2na-1-na^2}{1+na^2}\right)^2 \\ &= \frac{2\left(\left(na^2-1\right)^2 + \frac{1}{n}\left(-2na\right)^2\right)}{\left(1+na^2\right)^2} \\ &= \frac{2\left(n^2a^4 - 2na^2 + 1 + 4na^2\right)}{\left(1+na^2\right)^2} \\ &= \frac{2\left(n^2a^4 + 2na^2 + 1\right)}{\left(1+na^2\right)^2} \\ &= 2 \ . \end{aligned}$$

Secondly, A satisfies (2.23), as we now demonstrate. The quadratic function $q(x) := nx^2 - (n+1)x + 1$ has a positive leading coefficient (namely, n), and so, in the range $x \in \left[\frac{1}{n}, 1\right]$, attains its maximum at either $x = \frac{1}{n}$ or $x = 1$ (the maximum within this range occurs either at an endpoint of the range or at a point of inflection; the former must be the case since, as the leading coefficient is positive, the only point of inflection is a minimum). So, since $0 < b \leq 1$ (because, as we note above, b is non-zero and $(a, b, 0) \in R$) and $\frac{1}{ab} = n$ (because $(a, b, 0) \in G_n$), whence $a \geq \frac{1}{n}$, and since $a \leq 1$ (because $(a, b, 0) \in R$), we have that a is in the range $\left[\frac{1}{n}, 1\right]$, and so $q(a) \leq \max\left\{q\left(\frac{1}{n}\right), q(1)\right\} = 0$. That is,

$$na^2 - (n+1)a + 1 \leq 0 \ ;$$

multiplying by $2n$ and subtracting $n + n^2a^2$,

$$-2na + n + n^2a^2 - 2n^2a \leq -n - n^2a^2 \ . \tag{2.25}$$

So

$$\begin{aligned} \left(1 + na^2 - 2na\right)(1+n) &= 1 + na^2 - 2na + n + n^2a^2 - 2n^2a \\ &\overset{(2.25)}{\leq} 1 + na^2 - n - n^2a^2 \\ &= (1-n)\left(1 + na^2\right) \ . \end{aligned}$$

Hence, (because $n \in \mathbb{N}$, whence $1+n$ and $1+na^2$ are both positive), $\frac{1+na^2-2na}{1+na^2} \leq \frac{1-n}{1+n}$, and so, as claimed, A (of which the z-coordinate is $\frac{1+na^2-2na}{1+na^2}\sqrt{\frac{2}{n}}$) satisfies (2.23), as $\sqrt{\frac{2}{n}}$ is positive.

Thirdly, we consider (2.24). Recall that A has x-coordinate $\frac{2na^2}{1+na^2}$. Now, since $(a, b, 0) \in G_n$, whence $\frac{1}{ab} = n$, and since $(a, b, 0) \in R$, whence $a \leq b$, we have that

$$na^2 \leq nab = 1 \ .$$

Adding 1,

$$1 + na^2 \leq 2 = 2\left(1 + na^2\right) - 2na^2 \ .$$

Dividing by $1 + na^2$ (which is positive),

$$1 \leq 2 - \frac{2na^2}{1 + na^2} \ .$$

Hence, the point A satisfies (2.24), and so is on the curve of C_n. See the blue (■) elements of Fig. 2.11.

The projection A of an arbitrary point $(a, b, 0)$ on G_n from P_n onto the plane $y = 2 - x$ is on the curve of C_n.

Conversely, let $(c, 2-c, d)$ be an arbitrary point on C_n. By (2.22) (whence, if (x, y, z) is on C_n, then $z = \pm\sqrt{2 - 2(x-1)^2} + \sqrt{\frac{2}{n}} = \pm\sqrt{2x(2-x)} + \sqrt{\frac{2}{n}}$), we have that $d = \pm\sqrt{2c(2-c)} + \sqrt{\frac{2}{n}}$; by (2.23) (namely, that, if (x, y, z) is on C_n, then $z \leq \frac{1-n}{1+n}\sqrt{\frac{2}{n}}$, whence

$$z \leq 0 \tag{2.26}$$

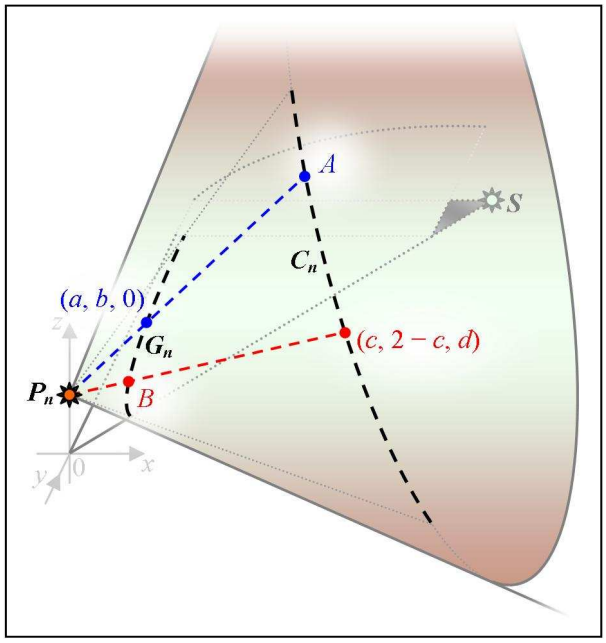

Figure 2.11: An illustration of the correspondence (via projection from P_n) between G_n and C_n, as described in Prop. 9.

since $n \geq 1$ gives that $\frac{1-n}{1+n}\sqrt{\frac{2}{n}} \leq 0$), we have that $d = -\sqrt{2c\,(2-c)} + \sqrt{\frac{2}{n}}$ (by (2.26), d is not positive, so we must take the negative root), so

$$(c, 2-c, d) = \left(c, 2-c, \sqrt{\frac{2}{n}} - \sqrt{2c\,(2-c)}\right) \quad .$$

The line that passes through both this point and $P_n = \left(0, 0, \sqrt{\frac{2}{n}}\right)$ is given by

$$\left\{ \begin{array}{rl} & \left(c, 2-c, \sqrt{\frac{2}{n}} - \sqrt{2c\,(2-c)}\right) \\ + & \gamma\left(c, 2-c, -\sqrt{2c\,(2-c)}\right) \end{array} \;\middle|\; \gamma \in \mathbb{R} \right\} \quad ;$$

that is, by

$$\left\{ \left((\gamma+1)\,c, (\gamma+1)\,(2-c), \sqrt{\frac{2}{n}} - (\gamma+1)\sqrt{2c\,(2-c)}\right) \;\middle|\; \gamma \in \mathbb{R} \right\} \quad .$$

This meets the plane $z = 0$ when $\gamma + 1 = \sqrt{\frac{2}{n}} \div \sqrt{2c\,(2-c)} = \sqrt{\frac{1}{nc(2-c)}}$, at the point $B := \left(\sqrt{\frac{c}{n(2-c)}}, \sqrt{\frac{2-c}{nc}}, 0\right)$, which we wish to show to be on G_n. Note that, by (2.23) (namely, if (x, y, z) is on the curve of C_n, then $z \leq \frac{1-n}{1+n}\sqrt{\frac{2}{n}}$), $\sqrt{\frac{2}{n}} - \sqrt{2c\,(2-c)} \leq \frac{1-n}{1+n}\sqrt{\frac{2}{n}}$, whence (by multiplying by -1 and adding $\sqrt{\frac{2}{n}}$)

$\sqrt{2c\,(2-c)} \geq \sqrt{\frac{2}{n}}\left(1-\frac{1-n}{1+n}\right) = \sqrt{\frac{2}{n}} \times \frac{2n}{1+n}$, so $2c\,(2-c) \geq \frac{2}{n} \times \frac{4n^2}{(1+n)^2} = \frac{8n}{(1+n)^2}$, whence $c\,(2-c) \geq \frac{4n}{(1+n)^2}$, and so $-c^2+2c-\frac{4n}{(1+n)^2} \geq 0$; from this, we have that

$$\frac{2}{n+1} \leq c \leq \frac{2n}{n+1} \tag{2.27}$$

(the negative leading coefficient of the quadratic $r\,(x) := -x^2 + 2x - \frac{4n}{(1+n)^2}$ means that r is non-negative inclusively between its two zeros, $x = \frac{2}{n+1}$ and $x = \frac{2n}{n+1}$, and nowhere else). Further, by (2.24) (that is, if (x, y, z) is on the curve of C_n, then $2 - x \geq 1$), $c \leq 1$, so

$$\frac{2}{n+1} \overset{(2.27)}{\leq} c \leq 1\ . \tag{2.28}$$

Therefore,

1. since c, n and $2-c$ are positive (because, respectively, of (2.27), the fact that $n \in \mathbb{N}$, and (2.28)), $0 \leq \sqrt{\frac{c}{n(2-c)}}$;

2. since $c \overset{(2.28)}{\leq} 1$, we have that $4c \leq 4$, whence $c^2 \leq 4 - 4c + c^2 = (2-c)^2$, whence $\frac{c}{n(2-c)} \leq \frac{2-c}{nc}$, and so $\sqrt{\frac{c}{n(2-c)}} \leq \sqrt{\frac{2-c}{nc}}$; and

3. since $\frac{2}{n+1} \overset{(2.27)}{\leq} c$, $2 \leq c\,(n+1)$, so $2 - c \leq nc$ and $\frac{2-c}{nc} \leq 1$, whence $\sqrt{\frac{2-c}{nc}} \leq 1$.

Hence, $0 \overset{1.}{\leq} \sqrt{\frac{c}{n(2-c)}} \overset{2.}{\leq} \sqrt{\frac{2-c}{nc}} \overset{3.}{\leq} 1$, and so $B = \left(\sqrt{\frac{c}{n(2-c)}}, \sqrt{\frac{2-c}{nc}}, 0\right) \in R$.

Further, we have that the product of the first and second coordinates of B is $\sqrt{\frac{c}{n(2-c)} \cdot \frac{2-c}{nc}} = \sqrt{\frac{1}{n^2}} = \frac{1}{n}$, and so the point is, as claimed, on G_n. See the red (■) elements of Fig. 2.11.

The projection B of an arbitrary point $(c, 2-c, d)$ on C_n from P_n onto the plane $z = 0$ is on the curve G_n.

This completes the proof. □

Remark 10. By Prop. 9, the radiation from P_n arriving at C_n passes through the curve in R corresponding (under the transformation described in Remark 7) to the curve $y = \frac{n}{x}$—that is, through G_n. The point on G_n through which such a ray passes corresponds, we recall from Prop. 8, to an *integer solution*[32] on this curve if and only if the point displays the interference pattern of S at maximum amplitude; that this is the case is then evident[33] at C_n.

[32]Recall that the correspondence is, in fact, to an integer solution (x, y) *where x and y are of the same parity*. Since we suppose n to be odd, however, all integer solutions on $y = \frac{n}{x}$ satisfy this condition (specifically, x and y are odd for all such solutions).

[33]The exact reasons for this evidence depend upon the implementations of sources S and P_n; see Remark 11.

Remark 11. We have not specified the exact nature of the type(s) of radiation emitted by S and P_n, instead mentioning in passing only the necessary property that radiation from S be transverse (recall point 4 of Definition 2), which has as a consequence wave activity orthogonal to the plane $z = 0$, which, we contrive, interferes with radiation from P_n in a way that is detectable at C_n. Suffice it to remark that *there exist* implementations resulting in such detectability; we need not—given the intention behind our introducing the factorization system, which intention does not include practical realization—describe fully such an implementation, though for illustration now outline an example.

Example 1. Suppose that S produces *water waves*, and P_n *visible light*, as follows. S regularly pounds the water's surface, producing waves that are reflected by barriers that model mirrors M_i, thus establishing an interference pattern with maximal wave activity at grid points and calmer waters elsewhere. Light from P_n shines through this unsettled surface, and a light sensor C_n (positioned so as to compensate for still-water refraction) detects

- steady light where rays have passed through (maximally) calm points on the water's surface, and
- intermittent light where they have passed through undulating points, resulting in periodically fluctuating refraction.

Points on C_n receiving the least light (summed over time) are those corresponding to grid points (and, hence, maximal choppiness and so maximal disruption of light from P_n due to refraction).

The exact details of this implementation are incidental to our discussion, though a rough illustration is given in Fig. 2.12.

Interpreting Results.

Remark 12. Recall from Remark 10 that the radiation arriving from P_n at a point on C_n will display maximal-amplitude interference (because of the standing-wave interference pattern from S) if and only if the point $(x, y, 0)$ of R through which the radiation passes offers a factorization of n (in that $\frac{1}{x} \cdot \frac{1}{y} = n$, where $\frac{1}{x}$ and $\frac{1}{y}$ are integers). Thus, the interpretation of results (i.e., the finding of factors) consists chiefly of conversion of the coordinates of a point on C_n (specifically, the point that displays diminution of radiation from P_n due to maximal-amplitude interference in the pattern of radiation from S) into those of a point in R (the point through which the ray passes). Proposition 10 describes this conversion.

Proposition 10. Radiation from P_n incident on a point $(c, 2-c, d)$ on C_n has passed through $\left(\sqrt{\frac{c}{n(2-c)}}, \sqrt{\frac{2-c}{nc}}, 0\right)$.

Proof. This claim is justified in the latter part of the proof of Prop. 9; recall also the red (■) elements of Fig. 2.11. □

Corollary 1. If the radiation from P_n arriving at $(c, 2-c, d)$ on C_n displays maximal-amplitude interference, then $\sqrt{\frac{nc}{2-c}}$ and $\sqrt{\frac{n(2-c)}{c}}$ are factors of n; conversely, each factor of n has a maximal-amplitude point so corresponding on C_n.

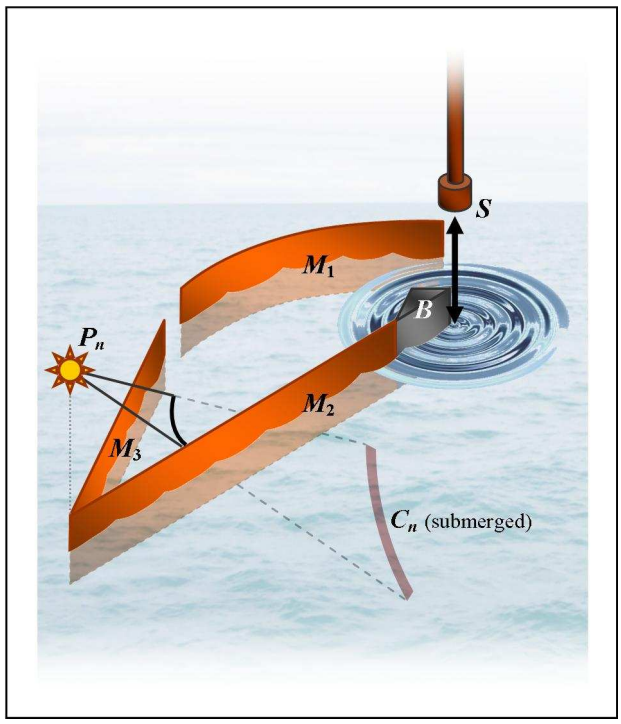

Figure 2.12: An implementation of the factorization system of Sect. 2.2, using water waves and visible light—see Example 1.

Proof. If radiation from P_n arriving at $(c, 2-c, d)$ displays maximal-amplitude interference, then, by Remark 10, the point on G_n through which the radiation has passed (which point is, by Prop. 10, $\left(\sqrt{\frac{c}{n(2-c)}}, \sqrt{\frac{2-c}{nc}}, 0\right)$) corresponds as per Remark 7 to an *integer solution* on $y = \frac{n}{x}$; recalling the transformation $\left(\frac{x}{n}, \frac{y}{n}, 0\right) \mapsto (x, y)$ of Remark 7, this integer solution is $\left(\sqrt{\frac{nc}{2-c}}, \sqrt{\frac{n(2-c)}{c}}\right)$. Hence, $\sqrt{\frac{nc}{2-c}}$ and $\sqrt{\frac{n(2-c)}{c}}$ are factors of n.

The "only if" of Remark 10 ensures that all factors of n give rise via the correspondence of Remark 7 to a point of maximal-amplitude interference on C_n. □

Remark 13. Given $(c, 2-c, d) \in \mathbb{R}^3$, it is possible *efficiently* to calculate $\sqrt{\frac{nc}{2-c}}$ and $\sqrt{\frac{n(2-c)}{c}}$ using a Turing machine: the 'difficult part' of finding factors of n has already been achieved once the position of the point $(c, 2-c, d)$ has been measured. See Sect. 2.2.3.

Remark 14. Having set up the system as described in Definitions 2 and 3, the factors of n are found as in Corollary 1. Since all factors are represented by points on C_n displaying maximal-amplitude interference (and since there are no other such points), a value of n induces

- no such points if and only if n is **not an integer**,[34]
- a single such point (corresponding to the factorization $n = 1.n$) if and only if n is **prime** (or **one**), and
- two or more such points (corresponding to the factorizations $n = f.\frac{n}{f}$ for all factors $f \leq \sqrt{n}$) if and only if n is **composite**.

In particular, one can envisage a method of identifying primes whereby the user sweeps continuously through a range of values of n (by continuously altering the wavelength of S, for example with a variable resistor, and the z-coordinate height of P_n and C_n (recall Remark 9)) whilst looking for the presence of exactly one point of maximal-amplitude interference.

Remark 15. Recall from Sect. 2.1.1 *Complexity of Factorization* that the security of the RSA cryptographic system relies upon the intractability of factorization. However, the ability of the proposed system to factorize quickly (albeit with efficiency hampered by other, non-time issues, as we discuss below) does not compromise this security: given technology sufficient to implement the system such that k-digit numbers may be reliably factorized (hence, we may decrypt information encoded with RSA using a k-digit key), with k maximal in this respect, we can, by Remark 14, efficiently find k-digit primes; by multiplying two such, we may form a $(2k-1)$- or $2k$-digit RSA key, which by maximality of k cannot be factorized by our system.

[34] Although the factorization problem demands that n be an integer—recall Sect. 2.1.1 *The Problems, Function and Decision*—, the means by which this input value is supplied to the system (namely, via alteration of the z-coordinate positioning of P_n and C_n, and of the wavelength of radiation from S) do not automatically preclude non-integer values.

Generalization.

Aside. We see above that the task of factorization has a geometric formulation as the extraction of integer solutions of an equation of which the graph is part of a conic section; this formulation is exploited by the factorization system described here.

As an aside, we note that, by repositioning P_n and C_n so as to implement a *different* cone, a method analogous to the one described here allows computation of integer solutions of *different* equations with conic-section (parabolic, hyperbolic, circular or elliptical) graphs. So, while factorization is chosen for discussion because of its wide range of applications and its notoriety as a difficult problem, the task is merely an illustration of a larger class of problems that the general method presented here solves (though not practicably—see Sect. 2.2.4).

2.2.2 Proof of Correctness

That the system does indeed correctly factorize its input value is the case by construction—see the preceding description of the system (Sect. 2.2.1)—; for clarity, we recap now the salient points.

Having set up the system as described in Definitions 2 and 3—including having set appropriately those aspects of the apparatus that depend upon the input value to be factorized, and, hence, having supplied this input value to the system—, we see from Corollary 1 that the sought factors are encoded[35] in the spatial coordinates of maximally dark points on a sensor (the measurement of which coordinates accordingly constitutes part of the system's output process).

This statement, whilst true, should certainly not be interpreted as suggesting that the system presented here offers an efficient or practicable method for factorizing large numbers. In particular, the two necessities (for the factorization to be correct) of

- the system's being set up as defined and
- spatial coordinates' being measured

carry with them a requirement of *precision*: only if these tasks are conducted with sufficient precision will the system function correctly; we discuss this issue further in Sect. 2.2.4.

The observation of Sect. 2.2.2, then, is merely that, *under the assumption that the system be set up exactly as described*, it will behave as described, and, in particular, produce the sought factors (albeit encoded in coordinates that must be measured). That is, *if the user is sufficiently precise* in his carrying out the input/output processes of the system, then the system will factorize correctly. This says nothing about the meaning of "sufficiently precise", nor about the *efficiency* with which the system factorizes (in terms of run-time, memory space, etc.): the system functions as claimed subject to these caveats re precision, but we have not thus far analyzed the system's *complexity* (nor, relatedly, have we quantified these caveats). We do this now.

[35] Decoding takes the form of simple calculations involving only subtraction, multiplication, division and the taking of radicals to bounded precision—see Sect. 2.2.3.

2.2.3 Time/Space Complexity

Using the System.

The use of the system to factorize $n \in \mathbb{N}$ consists of

1. calculation of the values $\frac{2}{n}$ (to be used as the wavelength of S) and $\sqrt{\frac{2}{n}}$ (to be used as the z-coordinate of P_n and of the centre of the circle of C_n);
2. supply of n to the system, by adjusting the wavelength of S and the z-coordinate of P_n and C_n in accordance with the values found during step 1;
3. interference of the radiation in the system, which entails propagation of the radiation over a fixed distance (since the same apparatus is, bar the adjustments made in step 2, used for any value of n);
4. measurement of the positions of maximal-amplitude interference points on the sensor C_n; and
5. conversion, via the mapping $(c, 2-c, d) \mapsto \left(\sqrt{\frac{nc}{2-c}}, \sqrt{\frac{n(2-c)}{c}}\right)$, of each position measured during step 4 into two factors of n.

We consider now the computational complexity of these five steps of the factorization system's use.

Time Complexity.

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

1. Though step 1 above entails calculation of a *fraction* and a *radical* (which, in general, requires arbitrary calculation time as the desired precision of the answer grows), we note that n is a natural number and that it is sufficient, assuming basic error correction[36], to calculate $\frac{2}{n}$ and $\sqrt{\frac{2}{n}}$ precisely enough only that n may be retrieved (given that $n \in \mathbb{N}$). Now, an approximation v to $\frac{2}{n}$ such that $v \in \left(\frac{4}{2n+1}, \frac{4}{2n-1}\right]$ has the property that $\left\lfloor \frac{2}{v} + \frac{1}{2} \right\rfloor = n$ (see Lemma 5); that is, n may be retrieved from such v. The interval $\left(\frac{4}{2n+1}, \frac{4}{2n-1}\right]$ has size $\frac{4}{2n-1} - \frac{4}{2n+1} = \frac{8}{4n^2-1} \in \mathcal{O}\left(n^{-2}\right)$; finding such v, then, requires calculation of (the order of) only twice as many decimal places of $\frac{2}{n}$ as n itself has digits. Similarly, $\sqrt{\frac{2}{n}}$ need be found only as an approximation in $\left(\sqrt{\frac{4}{2n+1}}, \sqrt{\frac{4}{2n-1}}\right]$, which requires calculation of (the order of) only $1\frac{1}{2}$ times as many decimal places of $\sqrt{\frac{2}{n}}$ as n has digits.

[36] Though supply to the system of n (via adjustment of the wavelength of S and positions of P_n and C_n) may be subject to noise, we may suppose that the system performs error correction, utilizing the fact that $n \in \mathbb{N}$. We defer exact details of this correction, commenting only that it may be achieved via standard techniques such as that described in Footnote 44 of Chap. 3.

Such computation, via standard, Turing-machine methods, requires run-time polynomial in the size of n (e.g., a variant on the Newton-Raphson method—see [80]—offers a quadratic-time approach).

2 – 4. Steps 2 – 4 take constant time: larger values of n take no longer to process in accordance with these stages. Notably, step 3, during which the actual factorization is performed, takes constant time—neither the propagation velocity of radiation from S nor the distance over which the radiation must propagate depends upon n—; compare this with known algorithmic methods, where computation time increases *exponentially* with the size of n (see, e.g., [41]).

5. Similarly to step 1, step 5 entails calculation of radicals though to only modest precision; specifically, $\sqrt{\frac{nc}{2-c}}$ and $\sqrt{\frac{n(2-c)}{c}}$ are integers, and so, as in step 1, calculation requires only quadratic run-time.

Thus, the time complexity of the system as a whole is, as claimed, *polynomial* in the size of the input.

Lemma 5. If $v \in \left(\frac{4}{2n+1}, \frac{4}{2n-1}\right]$ and $n \in \mathbb{N}$, then $\left\lfloor \frac{2}{v} + \frac{1}{2} \right\rfloor = n$.

Proof. If $\frac{4}{2n+1} < v \leq \frac{4}{2n-1}$, then $\frac{2n+1}{2} > \frac{2}{v} \geq \frac{2n-1}{2}$; i.e., $n - \frac{1}{2} \leq \frac{2}{v} < n + \frac{1}{2}$, whence the nearest integer $\left\lfloor \frac{2}{v} + \frac{1}{2} \right\rfloor$ to $\frac{2}{v}$ (rounding up when $\frac{2}{v}$ falls equidistant between consecutive integers) is n. □

Remark 16. Note from the above description of the system's use that steps 1 and 5 form a Turing-machine 'harness' that prepares input and output values (the former in readiness for supply to the system, the latter in readiness for comprehension by the system's user). This preparation takes polynomial time (as we see above) and space (see below), with the remaining three steps of the factorization process taking only constant time (see above) and space (see below).

Space Complexity.

Similarly, when considering the resource of *space*,[37] we see that only the Turing-machine calculations of steps 1 and 5 (which prepare input and interpret output as described in Remark 16) consume an increasing volume as n increases, and these only a *polynomially* increasing volume (in the size of the input)[38]; and for steps 2 – 4, the same, fixed-size apparatus—occupying the same, fixed space—is used for all values of n (though the positions of P_n and C_n depend upon n, there exists a finite, n-independent, bounding cuboid in which the apparatus lies *for all* n: each point of the apparatus lies in $[0,2] \times [0,2] \times \left[-\sqrt{2}, 2\sqrt{2}\right]$, which has volume $2 \times 2 \times 3\sqrt{2} = 12\sqrt{2}$ (this relies upon the inequality $\epsilon \leq \frac{\sqrt{3}-1}{2}$—recall Definition 2—, for else the maximal y-value taken by a point on M_1 is

[37] Space, as traditionally encountered with Turing machines, etc., can be viewed as the storage capacity of the memory required by a computation; we consider the analogous notion of required *physical volume*. This is formalized in the next chapter; see, in particular, Definition 20.

[38] These steps consist of *Turing-machine* processes, and we have, therefore, that the number of tape cells consumed is bounded above by the number of time steps elapsed—see Footnote 109 of Chap. 3.

$-\frac{1}{2(1+\epsilon)} + 2 + \epsilon > 2$)). Thus, the space complexity of the system is *polynomial* in the size of the input.[39]

Remark 17. The resources of time and space are arguably of paramount relevance when considering instances (Turing machines, random-access machines, etc.) of standard computational models. Notions of complexity developed with only these instances in mind, however, are understandably poor at capturing the complexity of instances of wildly different models; the factorization system above does indeed have polynomial time and space complexity[40], and yet *does* require exponentially increasing resource as n increases. Notably, larger values of n require *exponentially increasingly precise manipulation* of the input parameters (the wavelength of S and the positions of P_n and C_n) and *exponentially increasingly precise measurement* of the output parameters (the coordinates of points on C_n), and there is no reason for which we would not view *required precision* as a resource. Accordingly, we consider now (in an informal way, with formalization deferred to Sect. 3.3) the *precision complexity* of the system, which is certainly not polynomial, and hence better captures the system's true complexity than do the resources of time and space.

Aside. We note in passing that the system is qualitatively different from most existing factorization processes because it exploits a direct, physical implementation of the problem in preference to an instance of a standard computational model; this allows for much-improved calculation times, but time is not the only relevant resource. The suggestion here is that instances of different computational models may well consume different computational resources, which must be considered as part of a successful complexity analysis.

2.2.4 Precision Complexity

We see above that our analogue system for factorizing has both time and space complexity *polynomial* in the size (i.e., number of digits) of the number being factorized; as is remarked above, this is a pronounced improvement over the *exponential* run-time required by the best known Turing-machine approaches to factorization. One may wonder, then, what the catch is (for, of course, there is one); as alluded to above, the drawback with the system is related to *precision*.

We consider now the precision issues inherent in the use of the factorization system (we do this informally, formalizing in the next chapter—specifically, in Sect. 3.3—the concepts introduced here). The more general message is that

traditional complexity analysis (which considers the resources of time *and* space*, and variations thereon, but no others) is inadequate for capturing the true complexity of certain non-standard computers*

[39]Note that there is a conflict between size of apparatus and wavelength. The longest dimension of the apparatus is $3\sqrt{2}$, and the wavelength of S is $\lambda = \frac{2}{n}$; hence, for n of, say, 100 digits (whence $10^{99} \leq n < 10^{100}$), and λ as small as is practical (say that S produces γ-rays, with $\lambda = 10^{-10}$ m approximately), the longest side of the apparatus is between $\frac{3\times10^{89}}{\sqrt{2}}$ m and $\frac{3\times10^{90}}{\sqrt{2}}$ m, which is of the order of 10^{70} visible universe widths. Conversely, limiting the apparatus to a length of, say, 10 m, a wavelength of $\lambda \geq 10^{-10}$ m allows factorization of n at most $\left\lfloor \frac{10^{11}\sqrt{2}}{3} \right\rfloor$, which has eleven digits.

[40]This is in stark contrast with known factorization algorithms suitable for implementation via Turing machine, random-access machine or similar, of which the required time is *exponential* in the size of n—see [41].

(including our factorization system).

Motivation.

The intention of *precision complexity* is to capture the lack of robustness against input/output imprecision of a physical (analogue, optical, chemical, etc.) computer. This can be quantified by considering *real-number parameters* that characterize the imprecision in the system. Specifically (though, for now, informally), we may consider the space of tuples of these parameters, and the region therein of tuples for which a computation is successful (in that whatever errors are permitted by the parameters are corrected by the computing system); we define the *precision* of the computation to be one divided by the measure of this region.

We defer to Sect. 3.3 more formal and rigorous motivation for and definition of precision and precision complexity in the abstract case, and consider now the concrete example of precision in the context of the factorization system.

Factorization Example.

Consider the process of setting the wavelength λ of the source S.[41] Given the value $n \in \mathbb{N}$ to be factorized, the intention is to set λ to $\frac{2}{n}$. In practice (due, for example, to technological limitations on our control over the wavelength—we have in mind imprecise use of a variable resistor or similar), we may set the wavelength to $\frac{2}{n'}$, which we know only to lie in $\left[\frac{2}{n} - \epsilon, \frac{2}{n} + \epsilon\right]$ for some real-number error term $\epsilon \geq 0$.[42]

However, we suppose that the system corrects non-integer input values by rounding them so as to rectify 'small' errors[43]: given wavelength $\frac{2}{x}$ (for arbitrary $x \in \mathbb{R}$), the value to be factorized is taken to be the nearest integer $\left\lfloor x + \frac{1}{2} \right\rfloor$ to x (with $x + \frac{1}{2}$ taking precedence over $x - \frac{1}{2}$ when x is equidistant between two consecutive integers). So, provided that the supplied wavelength $\frac{2}{n'} \in \left[\frac{2}{n} - \epsilon, \frac{2}{n} + \epsilon\right]$ falls in the interval $\left(\frac{2}{n+\frac{1}{2}}, \frac{2}{n-\frac{1}{2}}\right]$ of values that the system successfully corrects to $\frac{2}{n}$—i.e., *provided that* $\left[\frac{2}{n} - \epsilon, \frac{2}{n} + \epsilon\right] \subseteq \left(\frac{2}{n+\frac{1}{2}}, \frac{2}{n-\frac{1}{2}}\right]$, *whence* $\epsilon < \frac{1}{n\left(n+\frac{1}{2}\right)}$—, then the system processes the correct input value.

So, for input value n, the set of corrigible errors ϵ in the input parameter λ is $\left[0, \frac{1}{n\left(n+\frac{1}{2}\right)}\right)$, which has measure (in this case, length) $\frac{1}{n\left(n+\frac{1}{2}\right)}$, and, hence, contributes to the system's precision complexity a multiplicative factor of $n\left(n + \frac{1}{2}\right)$; crucially, this factor increases *quadratically* with n, and hence *exponentially* with the size of n. Consideration of parameter λ alone, then, renders the system's precision complexity exponential, regardless of the contributions

[41] There are other input/output processes—the positioning of P_n and C_n and the measurement of points on C_n—that one may consider in a similar way, though, for our purposes, such consideration is unnecessary: once the setting of the wavelength has been shown (as, below, it is) to require exponentially increasing precision, then we have that the *overall* precision complexity is exponential, regardless of the (consequently redundant) contribution from other parameters.

[42] We have, then, an *additive error* in the wavelength; depending upon the details of a process's implementation, other forms of error are possible—some alternatives are discussed in Sects. 3.3.2 and 3.3.3.

[43] This error correction is discussed in Sect. 2.2.3 *Time Complexity* above.

of the other parameters (namely, the position of P_n and C_n and of points of maximal interference on C_n)—recall Footnote 41.

The analogue system's perceived time- and space-efficiency (present largely because of the system's directly, physically implementing the factorization problem rather than converting it into a contrived instance of the standard computation model[44]) serve to highlight and are testament to not the power of the system but rather the incompleteness of traditional complexity theory;

the situation exemplifies the inadequacy of traditional complexity theory for capturing the true complexity of non-Turing (for example, analogue) computers.

The system *does* require exponentially increasing resource (though neither specifically time nor space) as n increases: as n becomes greater, the user is required to position the apparatus and set a source's wavelength (so as to effect input) and measure the coordinates of points on the sensor (so as to obtain output) *with exponentially increasing precision*. Intuitively, then, the system has exponential *precision complexity* (this intuition is corroborated above and formalized in the following chapter). This suggests that the system's (exponential) precision complexity is of much greater significance than its (polynomial) time and space complexity. *The significant complexity measure is overlooked by traditional complexity theory*, which motivates the consideration of non-standard resources such as precision.

2.2.5 Summary

We now summarize Sect. 2.2.

In Sect. 2.2.1, we present an analogue system that factorizes natural numbers. The starting point for the system's derivation is the observation that the (numerical) search for a given number's factors can be recast as the (geometric) search for integer points on a certain curve. We go on to describe an implementation of both the grid of integer points and the above-mentioned curve (utilizing the fact that this curve is a conic section); the implementation is such that the intersection of the grid and the curve (which intersection consists of the sought integer points) can be identified using a sensor. We describe the way in which, given the coordinates of these sought points, factors of the input value may be recovered; this completes the description of the system itself.

In Sect. 2.2.2, we recap from Sect. 2.2.1 the results that demonstrate that the system does indeed correctly factorize its input value. We note, however, that the system has been shown to function as claimed only *in principle*: there are practical considerations—considerations, in particular, of the system's *computational complexity*—that should be made when assessing the system's viability if not its correct functioning.

In Sect. 2.2.3, we consider the system's time and space complexity, concluding that each is *polynomial* in the size of the input value (the factorization of n requires neither more physical space nor more computation time as n increases, other than as is required by the algorithmic 'harness' of Remark 16); this is a pronounced improvement over the known, exponential-time, polynomial-space

[44] Compare this comment with the view, mentioned in Sect. 2.2.1, of Susan Stepney.

algorithms[45] such as are discussed in [41]. We suggest in Sect. 2.2.3, however, that time and space alone do not capture the true complexity of the analogue system—that *precision*, rather, is the relevant resource.

In Sect. 2.2.4, we informally motivate and introduce the concept of *precision complexity* (which notion is formalized in the next chapter). We analyze the precision complexity of our analogue system, and conclude that it is *exponential* in the size of the system's input value. For the specific example of our analogue system, then, we have vindicated our claim that the true complexity is not captured by an analysis of time and space alone (is not captured, that is, by a *traditional* complexity analysis); more generally, we have demonstrated that certain types of computer warrant non-standard approaches to complexity theory—these approaches are developed and investigated throughout this dissertation.

2.3 Improved Analogue Factorization System

We discuss now an improved factorization system. The system was introduced by the author in [26, 27], on which papers much of Sect. 2.3 is based.

2.3.1 Relating the Original and Improved Systems

We see above (at least in an informal way) that the original analogue system is not an efficient means of factorizing; this is due to its prohibitive precision complexity. A major contributing factor to the precision of the system is its manufacturing process: slight inaccuracies in the physical construction of the apparatus can lead to wildly inaccurate results. In an attempt to alleviate this problem, we seek to replace the 'handmade' elements of the original system with 'automatic' components implemented via natural, physical phenomena, in a sense hopefully made apparent below.

One such drawback of the original system's design seems to be the 'artificial' implementation of the cone[46]—which implementation is little more than manual positioning of the cone's vertex (P_n) and a cross-sectional arc (C_n). *Direct* implementation of the cone by some physical phenomenon would, one expects, result in apparatus more precise than this 'handmade' system. We seek, therefore, a naturally occurring, precisely shaped cone.

Consider an omnidirectional burst, at time 0, of radiation from a point source in a plane P, and suppose that the radiation propagates at a constant velocity v, regardless of its direction or distance from the source (electromagnetic radiation,

[45] We emphasize that, in the present context, *algorithms* (that is, instances of 'algorithmic' models of computation: Turing machines, random-access machines, etc.) form a proper subset of *computers*, which latter set includes non-algorithmic, physical (e.g., analogue) devices.

[46] This drawback is crucial: a small imprecision in the shape/position of the cone leads to only a small imprecision in the resultant conic section, but this, in turn, leads to the system's reporting wildly incorrect 'factors', in part because a small perturbation in n can cause a large perturbation in the *factors of* n. Indeed, proper factors common to n and $n + a$ (for some small natural number a) are necessarily factors (greater than 1) of a itself—these factors are few since a is small; so, if n is subject to some small imprecision and becomes corrupted as $n+a$, then the resultant, incorrect output (i.e., the factors of $n+a$) will have little in common with the desired, correct output (factors of n). In particular, n and $n + 1$ share no proper factors, and so entering n with an additive error of 1 renders the resultant output multiset entirely incorrect (i.e., disjoint with the sought multiset).

for example, satisfies this criterion). The points reached by the radiation at time $t \geq 0$ describe a circle of radius vt (see Fig. 2.13(a)). Regarding this increasing circle in the three-dimensional space with two spatial axes (those of P) and one temporal axis, we have a perfectly constructed cone (see Fig. 2.13(b)).

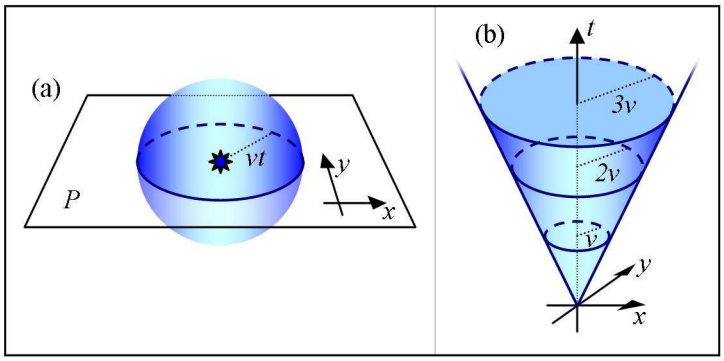

Figure 2.13: (a) The circle of radiation in plane P at time $t \geq 0$. (b) The conic structure of the circle as plotted against t.

The original implementation of the cone (as described in Sect. 2.2.1), which we seek to replace with this burst of radiation, exists in three spatial dimensions; relative to this, the proposed replacement is effectively rotated so as to span two spatial dimensions and one temporal. We now rotate the grid—in the intersection of which with the cone we are interested—similarly. The plane, Q, say, of the grid must lie parallel to and positively distant from the cone's axis, so that the resultant conic section is the required hyperbola; accordingly, in the radiation-burst implementation, plane Q is modelled as a permanent line in plane P, positively distant from the point source (Q thus spans one spatial and one temporal dimension). Within Q, the grid itself consists of points equally spaced along both axes: existing instantaneously at regular time intervals, and at regularly spaced points along the spatial axis.

Whilst the principle is as with the original analogue system—we seek points of intersection of a cone and a planar grid—, then, the implementation is different: before, the cone and grid existed in three-space; now, they exist (time-dependently) in the plane, the cone as a steadily increasing circle, the grid as a flashing row of points. Further, since we seek points of intersection with the grid, we need consider only those instants when the row of grid points is 'on', at which instants (together) the cone is modelled as a family of concentric circles with arithmetically progressional radii;

we seek the intersection of a row of points and a nest of concentric circles.

The improved, analogue factorization system, of which the design is motivated by the informal comments of Sect. 2.3.1, is now formally described.

2.3.2 Description

Apparatus.

We describe the apparatus as lying in a plane; a physical realization would see the relevant features (namely, X, Y, a and b) extended so as to have thickness

in the third dimension, though with readings being taken, etc. within the plane.

We begin with a provisional definition. In light of the discussion immediately following the definition, it is modified to give Definition 5.

Definition 4 (provisional; see Definition 5)**.**

- Let n be the natural number to be factorized.[47]
- Let X be an opaque screen occupying the line $(\mathbb{R} \setminus \{0, 2\sqrt{n}\}) \times \{0\}$. The breaks at $(0, 0)$ and $(2\sqrt{n}, 0)$ in this line model slits in the screen; call these slits a and b respectively.
- Let S be a source at $(\sqrt{n}, -\sqrt{n})$ of radiation[48] of wavelength 1.
- Let Y be a screen occupying the line $\{0\} \times \mathbb{R}^+$.[49] The negligible width of slit a in X lies to the $x > 0$ side of Y.

Note (and this is the point that prompts us to modify our definition) that the layout of the apparatus—specifically the distance separating a and b, and the position of S—depends upon n. So as to be able to use the same apparatus to factorize different values, then, we instead scale each axis in the plane by a factor of $\frac{1}{2\sqrt{n}}$;[50] accordingly, we replace the previous definition with the following.

Definition 5 (to replace Definition 4)**.**

- Let n be the natural number to be factorized.
- Let X be an opaque screen occupying the line $(\mathbb{R} \setminus \{0, 1\}) \times \{0\}$. The breaks at $(0, 0)$ and $(1, 0)$ in this line model slits in the screen; call these slits a and b respectively.
- Let S be a source at $\left(\frac{1}{2}, -\frac{1}{2}\right)$ of radiation of wavelength $\frac{1}{2\sqrt{n}}$.
- Let Y be a screen occupying the line $\{0\} \times \mathbb{R}^+$. The negligible width of slit a in X lies to the $x > 0$ side of Y.

See Fig. 2.14.

In so replacing Definition 4, we may reuse the same apparatus to factorize any natural-number value n, having to change only the wavelength of S rather than the system's physical layout.

The intention of this apparatus's use—which we illustrate in Fig. 2.15 for the example $n = 15$—is to observe radiation incident on Y from S via a and b (we see in Prop. 12 the significance in terms of the factorization of n of the positions of certain aspects of this radiation).

[47] With the previous system (see Sect. 2.2), we assume for convenience that n is odd; we make no such assumption here.

[48] As with the previous system, we do not wish to constrain particularly stringently the type of radiation; we require of it only that its waves, upon meeting, exhibit interference in the normal way. So as to demonstrate existence of suitable radiation types, we note that *electromagnetic* radiation is sufficient (but by no means necessary) for our purposes.

[49] We define $\mathbb{R}^+$ to be the set $[0, \infty)$ of non-negative real numbers.

[50] It is this flavour of scaling operation that gives rise to a trade-off that exists between *precision* and *space* complexity; this is discussed in Sect. 3.8.1 *Precision and Space*.

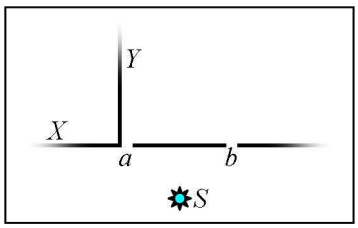

Figure 2.14: The apparatus described in Definition 5.

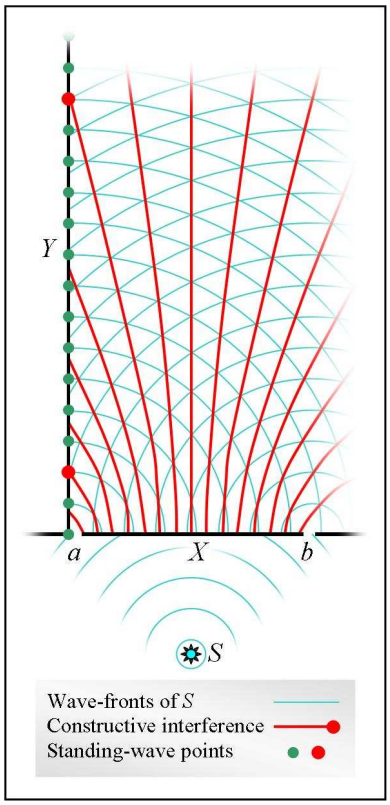

Figure 2.15: An example (in which $n = 15$) of the use of the apparatus described in Definition 5.

- The *wave-fronts* of radiation from S are shown in Fig. 2.15 in blue (■)—for our purposes, these are little more than construction lines to aid in locating points of constructive interference.
- Red (■) lines/points in Fig. 2.15 are those of *constructive interference* in the interference pattern caused by the radiation from S having passed through a double-slit arrangement (consisting of a and b). Such points are, by the nature of interference, those of which the respective distances from a and from b differ by an integer multiple of the wavelength $\frac{1}{2\sqrt{n}}$ of radiation from S; these points are the elements of the set

$$\left\{ (x,y) \in \left(\mathbb{R}^+\right)^2 \;\middle|\; 2\sqrt{n}\left[\left(x^2+y^2\right)^{\frac{1}{2}} - \left(\left(x-1\right)^2+y^2\right)^{\frac{1}{2}}\right] \in \mathbb{Z} \right\} \quad .$$

- Solid circles of either green or red (■/■) along screen Y represent points of maximal wave activity in a *standing wave* that we suppose to have been instantiated along Y; the points of maximal wave activity—of which one coincides with the origin of our coordinate system—are at S-wavelength spacing, and, therefore, form the set $\left\{ \left(0, \frac{k-1}{2\sqrt{n}}\right) \in \mathbb{R}^2 \;\middle|\; k \in \mathbb{N} \right\}$.

It is true (though certainly not a priori obvious; we prove the fact in Prop. 11) that there exist points displaying both

- constructive interference in the double-slit interference pattern from S and
- maximal wave activity in the standing wave along Y.

Call such points *M-points* (because of the *m*aximality of wave activity at these points).

In the example of Fig. 2.15 in which $n = 15$, there are two M-points—shown as red (■) circles—at $\left(0, \frac{1}{\sqrt{n}}\right)$ and $\left(0, \frac{7}{\sqrt{n}}\right)$.

At M-points, maximality of the constituent (standing-wave and double-slit-pattern) addends gives maximality of the resultant, total wave activity; this fact is exploited in order to identify M-points, and, as we see below, from M-points' positions can be retrieved the factors of n.

Let E be an M-point. Since E exhibits maximal standing-wave activity, its distance from a is an integer multiple of $\frac{1}{2\sqrt{n}}$. Further, since E exhibits constructive slit-pattern interference, its respective distances from a and from b differ by a multiple of $\frac{1}{2\sqrt{n}}$; since, by the previous sentence, the former distance is a multiple of $\frac{1}{2\sqrt{n}}$, so must be the latter.

E is therefore both on one of the family of circles of which the radii are multiples of the wavelength of S and with centre b, and coincident with a standing-wave (i.e., S-wavelength-multiple) point on the y-axis. This implementation, then, allows identification (via their maximal brightness) of the sought points described informally in Sect. 2.3.1 (we seek, recall, the intersection of a row of points and a nest of concentric circles).

Input to the System.

As alluded to immediately after Definition 5, the input value n is supplied to the system by altering to $\frac{1}{2\sqrt{n}}$ the wavelength of the radiation from S. All other

aspects of the system's physical structure are independent of the input value. (In the example of Fig. 2.15, then, the input value of 15 is conveyed to the system by the user's adjusting to $\frac{1}{2\sqrt{15}} = 0.129\ldots$ the wavelength of radiation from S.)

Since, by definition of the factorization problem, n is known to be a natural number, we may suppose that the system effects error correction:[51] given a wavelength $\frac{1}{2\sqrt{n'}}$ corresponding to a *non-integer* input value $n' \in \mathbb{R}$, the system takes the input value with which it computes to be the nearest integer $\left\lfloor n' + \frac{1}{2} \right\rfloor$ to n' (rounding upwards when n' is equidistant between two consecutive integers). Consequently, in order successfully to input the intended value n, it is sufficient (and necessary[52]) to set the wavelength of S to a real-number value in $I_n := \left(\frac{1}{2\sqrt{n+\frac{1}{2}}}, \frac{1}{2\sqrt{n-\frac{1}{2}}} \right]$ (since members of I_n, and only these members, get corrected to n).

Note that the operations (extraction of square root, multiplication and division) used in calculating the wavelength $\frac{1}{2\sqrt{n}}$ given input value n can be performed by a Turing machine in time and space polynomial in the size $\log n$ of n: there is no 'sleight of hand' whereby costly calculation is swept under the carpet of the input/output processes or otherwise tacitly assumed to come for free. Strictly speaking, the time/space required to calculate a square root depends not only upon the size of the radicand, but also upon the precision with which the answer is to be supplied; specifically, the Newton-Raphson method reduces the time and space complexity of finding a square root to k digits of precision to that of multiplying two k-digit numbers (see [80]), which, by long multiplication for example, is achievable in $\mathcal{O}\left(k^2\right)$ time and, hence, space. In our case, bounded precision is sufficient since we need only calculate a value in I_n: the wavelength need be found only with sufficient precision that n can be recovered thence.

We ask, then, how many digits of precision are needed. Suppose that we compute $\sqrt{n}$ to k-digit precision (and suppose, without loss of generality, that we work in base 10) where k is a natural number not less than $\log_{10} 4 + \frac{1}{2}\log_{10}\left(n + \frac{1}{2}\right) \in \mathcal{O}(\log n)$;[53] then our computed value—r, say—for $\sqrt{n}$ lies in the interval $\left[\sqrt{n} - \frac{1}{10^k}, \sqrt{n} + \frac{1}{10^k}\right]$.[54]

We claim that this results in the corresponding wavelength value $\frac{1}{2r}$ (which lies in $J_{n,k} := \left[\frac{1}{2\left(\sqrt{n}+10^{-k}\right)}, \frac{1}{2\left(\sqrt{n}-10^{-k}\right)} \right]$) lying in I_n, and, hence, results in the correct factorization's being performed. We have that $k \geq \log_{10}\left(4\sqrt{n+\frac{1}{2}}\right)$, so (by isotonicity of logarithms) $10^k \geq 4\sqrt{n+\frac{1}{2}}$, whence $\frac{1}{10^k} \leq \frac{1}{4\sqrt{n+\frac{1}{2}}}$, which,

[51] Cf. Footnotes 36 of this chapter and 44 of Chap. 3.

[52] Here we see the root of the input process's precision complexity.

[53] Note that we do not write '$\mathcal{O}(\log_{10} n)$', including the base, here for reasons given in Footnote 5.

[54] That this interval is closed rather than open (i.e., that it includes its endpoints) reflects the fact that we permit the decimal expansions of $\sqrt{n}$ and of r to end with infinite strings of 9s; whether we allow this is unimportant in determining (up to $\mathcal{O}$-notation) which values of k are sufficiently large that the computation proceeds correctly.

along with Lemma 6 below, gives that

$$\frac{1}{10^k} < \sqrt{n+\frac{1}{2}} - \sqrt{n} \ ; \tag{2.29}$$

by adding $\sqrt{n}$, then, $\sqrt{n} + \frac{1}{10^k} < \sqrt{n+\frac{1}{2}}$, whence, by taking reciprocals and halving,

$$\frac{1}{2\left(\sqrt{n}+\frac{1}{10^k}\right)} > \frac{1}{2\sqrt{n+\frac{1}{2}}} \ . \tag{2.30}$$

Further, by (2.29) and Lemma 7 below, $\frac{1}{10^k} < \sqrt{n} - \sqrt{n-\frac{1}{2}}$, whence, by adding $\sqrt{n-\frac{1}{2}} - \frac{1}{10^k}$, $\sqrt{n-\frac{1}{2}} < \sqrt{n} - \frac{1}{10^k}$, and so, by taking reciprocals and halving,

$$\frac{1}{2\sqrt{n-\frac{1}{2}}} > \frac{1}{2\left(\sqrt{n}-\frac{1}{10^k}\right)} \ . \tag{2.31}$$

Therefore, we have that $\frac{1}{2r} \in J_{n,k} \overset{(2.30,\ 2.31)}{\subseteq} I_n$ (recall that $J_{n,k}$ is the interval $\left[\frac{1}{2\left(\sqrt{n}+10^{-k}\right)}, \frac{1}{2\left(\sqrt{n}-10^{-k}\right)}\right]$ and I_n the interval $\left(\frac{1}{2\sqrt{n+\frac{1}{2}}}, \frac{1}{2\sqrt{n-\frac{1}{2}}}\right]$). This k, which is in $\mathcal{O}(\log n)$, represents sufficient precision in calculating square roots that factorization is performed correctly.

Note that the size $|I_n|$ of I_n is $\frac{1}{2\sqrt{n-\frac{1}{2}}} - \frac{1}{2\sqrt{n+\frac{1}{2}}} = \frac{\sqrt{n+\frac{1}{2}}-\sqrt{n-\frac{1}{2}}}{2\sqrt{n^2-\frac{1}{4}}}$. We claim that

$$\frac{1}{4n\sqrt{n}} < |I_n| < \frac{1}{4\left(n-\frac{1}{2}\right)\sqrt{n+\frac{1}{2}}} \ . \tag{2.32}$$

The former inequality holds because $n^2 - \frac{1}{4} + \frac{1}{64n^2} > n^2 - \frac{1}{4}$, i.e., $\left(n-\frac{1}{8n}\right)^2 > \left(\sqrt{n^2-\frac{1}{4}}\right)^2$, whence (by isotonicity of non-negative squaring) we have that $n - \frac{1}{8n} > \sqrt{n^2-\frac{1}{4}}$, so (adding $\frac{1}{8n} - \sqrt{n^2-\frac{1}{4}}$) $n - \sqrt{n^2-\frac{1}{4}} > \frac{1}{8n}$, whence $2n - 2\sqrt{n^2-\frac{1}{4}} > \frac{1}{4n} = \left(\frac{1}{2\sqrt{n}}\right)^2$; this left-hand side is equal to $\left(n+\frac{1}{2}\right) - 2\sqrt{n^2-\frac{1}{4}} + \left(n-\frac{1}{2}\right) = \left(\sqrt{n+\frac{1}{2}} - \sqrt{n-\frac{1}{2}}\right)^2$, so (again by isotonicity of squaring) $\frac{1}{2\sqrt{n}} < \sqrt{n+\frac{1}{2}} - \sqrt{n-\frac{1}{2}}$. The latter inequality holds for similar reasons.[55]

We return to (2.32) in Sect. 2.3.7.

Lemma 6. With n as in the above discussion, $\frac{1}{4\sqrt{n+\frac{1}{2}}} < \sqrt{n+\frac{1}{2}} - \sqrt{n}$.

Proof. $\left(n+\frac{1}{4}\right)^2 = n^2 + \frac{1}{2}n + \frac{1}{16} > n^2 + \frac{1}{2}n = \left(\sqrt{n\left(n+\frac{1}{2}\right)}\right)^2$, so (by isotonicity of squaring for non-negative arguments) $n + \frac{1}{4} > \sqrt{n\left(n+\frac{1}{2}\right)}$, whence (by adding

[55] We leave explication of the details to the reader, though note that the key observation is that $\sqrt{n+\frac{1}{2}} - \sqrt{n-\frac{1}{2}} < \frac{1}{2\sqrt{n-\frac{1}{2}}}$.

$\frac{1}{4} - \sqrt{n\left(n+\frac{1}{2}\right)}$) we have that $n + \frac{1}{2} - \sqrt{n\left(n+\frac{1}{2}\right)} > \frac{1}{4}$, which, by dividing by $\sqrt{n+\frac{1}{2}}$, yields the sought result. □

Lemma 7. With n as in the above discussion, $\sqrt{n+\frac{1}{2}} - \sqrt{n} < \sqrt{n} - \sqrt{n-\frac{1}{2}}$.

Proof. $n^2 > n^2 - \frac{1}{4}$, so $n = \sqrt{n^2} > \sqrt{n^2 - \frac{1}{4}}$ (by isotonicity of taking square roots), and $2n > 2\sqrt{n^2 - \frac{1}{4}}$. Hence (adding $2n$), $4n > 2n + 2\sqrt{n^2 - \frac{1}{4}} = n + \frac{1}{2} + 2\sqrt{n+\frac{1}{2}}\sqrt{n-\frac{1}{2}} + n - \frac{1}{2} = \left(\sqrt{n+\frac{1}{2}} + \sqrt{n-\frac{1}{2}}\right)^2$, so $2\sqrt{n} > \sqrt{n+\frac{1}{2}} + \sqrt{n-\frac{1}{2}}$ (by isotonicity of squaring). Rearranging gives the sought result. □

Output from the System.

Having set up the apparatus as described in Definition 5 (including provision to the system of n, encoded in the wavelength of S), an interference pattern is produced on screen Y (since, on the $y > 0$ side of X, a and b act as separate, mutually coherent sources,[56] of which the respective waves interfere).

Since effective sources a and b are in phase, a point E on Y exhibits full, constructive interference (and, hence, maximal brightness) if and only if the respective distances from E to a and from E to b are integer multiples of the wavelength $\frac{1}{2\sqrt{n}}$ (recall the standing wave of Sect. 2.3.2 *Apparatus*); that is, if and only if E is an M-point. We show now that such points exist.

Proposition 11. This maximal brightness is attained.

Proof. Let $E = \left(0, \frac{n-1}{2\sqrt{n}}\right)$. The distances from E to a (which lies at $(0,0)$) and from E to b (at $(1,0)$) are, respectively, $\frac{n-1}{2\sqrt{n}}$ and

$$\begin{aligned}\sqrt{1^2 + \left(\frac{n-1}{2\sqrt{n}}\right)^2} &= \sqrt{1 + \frac{n^2-2n+1}{4n}} \\ &= \sqrt{\frac{n^2+2n+1}{4n}} \\ &= \frac{n+1}{2\sqrt{n}}\ .\end{aligned}$$

Each of these distances is an integer multiple of $\frac{1}{2\sqrt{n}}$; hence, E is an M-point, exhibiting full, constructive interference. □

The process whereby output is read from the system consists of

- identifying a maximally bright (i.e., M-) point E on Y,
- measuring the y-coordinate h of E, and
- calculating the values $p := \sqrt{n}\left(\sqrt{h^2+1} + h\right)$ and $\frac{n}{p}$.

In Fig. 2.15 (where $n = 15$), for example,

[56] This coherence is because S lies on the perpendicular bisector $x = \frac{1}{2}$ of ab.

- there are M-points on Y at $\left(0, \frac{1}{\sqrt{15}}\right)$ and $\left(0, \frac{7}{\sqrt{15}}\right)$,
- which have respective y-coordinates of $\frac{1}{\sqrt{15}}$ and $\frac{7}{\sqrt{15}}$,
- corresponding to respective p-values of $\sqrt{15}\left(\sqrt{\left(\frac{1}{\sqrt{15}}\right)^2+1}+\frac{1}{\sqrt{15}}\right)=5$ and $\sqrt{15}\left(\sqrt{\left(\frac{7}{\sqrt{15}}\right)^2+1}+\frac{7}{\sqrt{15}}\right)=15$ (and $\frac{n}{p}$-values of 3 and 1).

As is proven in Prop. 12, the values p and $\frac{n}{p}$ corresponding in this way to any M-point *are factors of* n (and, since $h \geq 0$, we have that $\frac{n}{p} \leq \sqrt{n} \leq p$). Further, each factor of n that is at least $\sqrt{n}$ occurs as such p, and each factor that is at most $\sqrt{n}$ as $\frac{n}{p}$, for some M-point. Thus, by processing all such points as described here, all factors of n are found.

As during input, there is no tacit presumption of a (Turing-) computationally complex operation: the process of finding p and $\frac{n}{p}$ given h and n is algorithmically efficient (e.g., via the Newton-Raphson method—see [80]).

2.3.3 Proof of Correctness

Proposition 12. A point E on Y is maximally bright (i.e., is an M-point) if and only if the corresponding value of p (namely, $\sqrt{n}\left(\sqrt{h^2+1}+h\right)$, where h is the y-coordinate of E) is a factor of n no less than $\sqrt{n}$ (and, hence, $\frac{n}{p}$ a factor at most $\sqrt{n}$).

Proof. Suppose that $E := (0, h)$ is maximally bright. Then the respective distances—h and $\sqrt{h^2+1}$—from E to a and from E to b are integer multiples of $\frac{1}{2\sqrt{n}}$; that is, $\alpha := 2\sqrt{n}h$ and $\beta := 2\sqrt{n}\sqrt{h^2+1}$ are integers (as, hence, are α^2 and β^2). Now

$$\beta^2 - \alpha^2 = 4n\left(h^2+1\right) - 4nh^2 = 4n \ , \tag{2.33}$$

which is even, so α^2 and β^2 are integers of the same parity; therefore, α and β are integers of the same parity. Hence, both $\beta \pm \alpha$ are even, and so both $\frac{\beta \pm \alpha}{2}$ are integers, of which the product is $\frac{\beta+\alpha}{2} \cdot \frac{\beta-\alpha}{2} = \frac{\beta^2-\alpha^2}{4} \overset{(2.33)}{=} n$; that is, both $\frac{\beta \pm \alpha}{2}$ are factors of n. Now $p := \sqrt{n}\left(\sqrt{h^2+1}+h\right)$ is precisely $\frac{\beta+\alpha}{2}$, and is, therefore, a factor of n. Further, where $q = \sqrt{n}\left(\sqrt{h^2+1}-h\right)$, the product pq is $n\left(\sqrt{h^2+1}+h\right)\left(\sqrt{h^2+1}-h\right) = n\left(h^2+1-h^2\right) = n$, and $p \geq q$, so $p \geq \sqrt{n}$, as required.

Conversely, suppose that the point $E := (0, h)$ on Y is such that $p := \sqrt{n}\left(\sqrt{h^2+1}+h\right)$ is a factor of n (we have, further, that $p \geq \sqrt{n}$ since $h \geq 0$). Again, let $q = \sqrt{n}\left(\sqrt{h^2+1}-h\right)$; then, as in the previous paragraph, $pq = n$, and so (since, by hypothesis, $p \mid n$) $q \in \mathbb{Z}$. We have that p and q are integers; so too, then, are their sum $p+q = 2\sqrt{n}\sqrt{h^2+1}$ and difference $p-q = 2\sqrt{n}h$; that is, $\sqrt{h^2+1}$ and h—which are the respective distances from E to b and from E to a—are integer multiples of $\frac{1}{2\sqrt{n}}$. Thus, E is maximally bright, as required. □

Having set up the system as in Definition 5, including having input n (encoded in the wavelength of the radiation from S), the factors of n are found by measuring the y-coordinates of M-points on Y and converting these into the corresponding values p and $\frac{n}{p}$; Prop. 12 guarantees that the output values so produced from all M-points are the factors of n and only the factors of n.

2.3.4 Practical Considerations

The description given here of the system is an abstraction of any physical realization thereof: aspects of the description require modification before the system can be implemented in practice. We note at the beginning of Sect. 2.3.2 *Apparatus* that the confinement of the system's structure to a plane is one such aspect; thus, the screens X and Y, and slits a and b, should actually have positive, z-axis height, while S should remain as much as is practicable a point source, in the plane of which measurements (specifically of M-points' positions) on Y should be taken.

Further, we cannot physically realize the infinitely long screen X. However, since we require of X and S only that, on the $y > 0$ side of X, a and b act as mutually coherent sources, we may replace X with a finite box X' occupying the set

$$\begin{aligned} & \left(\left\{ (x,0) \in \mathbb{R}^2 \;\middle|\; -1 \leq x \leq 2 \right\} \setminus \{(0,0),(1,0)\}\right) \\ \cup \quad & \left\{ (x, x-2) \in \mathbb{R}^2 \;\middle|\; \frac{1}{2} \leq x \leq 2 \right\} \\ \cup \quad & \left\{ (x, -x-1) \in \mathbb{R}^2 \;\middle|\; -1 \leq x \leq \frac{1}{2} \right\} \quad , \end{aligned}$$

sealed but for slits a and b, and containing S (which retains its position, as do a and b); we assume the internal surfaces of X' to be non-reflective, so that radiation from S that does not directly reach a or b is absorbed. (Of course, X' has a positive, z-axis height in light of the preceding paragraph.) Figure 2.16 shows the apparatus after this and the following modification.

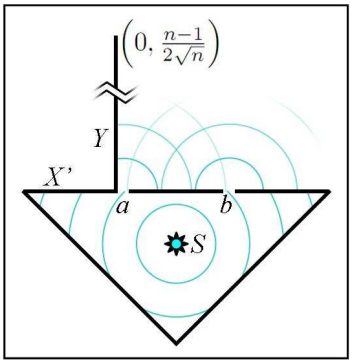

Figure 2.16: The apparatus as modified in Sect. 2.3.4. Wave-fronts of radiation from S are shown (in blue (■)) to illustrate the double-slit nature of interference in the region $x, y \geq 0$ above X' and to the right of Y.

In the same vein, we cannot realize the infinitely long screen Y. It suffices,

though, for Y to occupy the section of the y-axis between $y = 0$ and $y = \frac{n-1}{2\sqrt{n}}$.[57] This is because of the following proposition.

Proposition 13. The y-coordinate—h, say—of any maximally bright point on Y satisfies $h \in \left[0, \frac{n-1}{2\sqrt{n}}\right]$.

Proof. By Prop. 12, the M-point $(0, h)$ corresponds to a pair $\left(p, \frac{n}{p}\right)$ of factors of n (where $p \geq \sqrt{n} \geq \frac{n}{p}$—note in particular, then, that

$$p \in \left[\sqrt{n}, n\right] \quad , \tag{2.34}$$

since factors of n cannot exceed n); explicitly, we recall from Sect. 2.3.2 *Output from the System*, the correspondence is that $p = \sqrt{n}\left(\sqrt{h^2+1} + h\right)$.

First, we express h in terms of p and n.

$$\sqrt{h^2+1} + h = \frac{p}{\sqrt{n}} \quad , \tag{2.35}$$

whence (by squaring) $2h^2 + 1 + 2h\sqrt{h^2+1} = \frac{p^2}{n}$, so (by subtracting 1) $\frac{p^2-n}{n} = 2h\left(h + \sqrt{h^2+1}\right) \overset{(2.35)}{=} \frac{2hp}{\sqrt{n}}$, and so

$$h = \frac{p^2 - n}{2\sqrt{n}p} \quad .$$

In particular,

$$\text{if } p = \sqrt{n}, \text{ then } h = 0, \text{ whereas if } p = n, \text{ then } h = \frac{n-1}{2\sqrt{n}}. \tag{2.36}$$

For fixed n, h as a function of p (i.e., $h\colon p \mapsto \frac{p^2-n}{2\sqrt{n}p}$) is *isotone* (this is perhaps more readily evident from p expressed in terms of h—$p = \sqrt{n}\left(\sqrt{h^2+1} + h\right)$—than vice versa), so from values of p between $\sqrt{n}$ and n (such values are guaranteed by (2.34)) we obtain values of h between the corresponding values (given in (2.36)) 0 and $\frac{n-1}{2\sqrt{n}}$: $h \in \left[0, \frac{n-1}{2\sqrt{n}}\right]$, as required. □

Thus, we may truncate at $y = \frac{n-1}{2\sqrt{n}}$ the screen Y (see again Fig. 2.16) without losing any M-points, and, hence, without missing any factors of n when such points' positions are converted into factors.

(One practical consideration not discussed in the context of the present section's factorization system concerns the production of radiation of sufficiently short wavelength. In particular, as n increases, the required wavelength of S corresponds to an increasingly impractical amount of energy. We see below, however, that other considerations—notably, that of *precision*—render the system impracticable; whilst these concerns about wavelength are certainly not *misplaced*, then, they are at least *redundant*. Further, we treat energy in other contexts—see Sects. 3.6 and 3.8.1 *Energy and Time*, for example.)

[57] In making this observation, we solve the problem of having to implement the infinite screen Y, but reintroduce the undesirable property of the system's layout's depending upon n. This, we see below, renders the system unsuitable for factorizing arbitrary natural numbers, but is not a problem when factorizing numbers satisfying certain criteria; it is common, furthermore, for public keys used in the RSA cryptographic system to satisfy these criteria.

2.3.5 Time/Space Complexity

Time Complexity.

Consider first the *time complexity* of the system. The operation of the system consists of

1. the provision to the system of the value n to be factorized, by way of adjustment of the wavelength of source S;
2. the 'processing' by the system of the input value, whereby an interference pattern is produced on Y;[58]
3. the measurement of the y-coordinates of M-points on Y; and
4. the conversion of these coordinates into factors of n.

Of these, only stages 1 and 4 take longer as n increases—because, respectively, of the Turing-machine-realm processing of n to find the corresponding wavelength $\frac{1}{2\sqrt{n}}$ (see Sect. 2.3.2 *Input to the System*), and of y-coordinates to find factors (see Sect. 2.3.2 *Output from the System*)—, and even these take only polynomially long in the size $\log n$ of n; thus, the processing stages (2 and 3), which lie in the physical-computing realm, have *constant* time complexity, and their algorithmic 'harness', which prepares input and interprets output, has *polynomial* time complexity (as does the system as a whole, therefore).[59]

Space Complexity.

Consider now the system's *space complexity*. The apparatus has

- negligible, constant height (along the z-axis),
- a constant width (along the x-axis) of three (due to X'), and
- a depth (along the y-axis) of $\frac{n+3\sqrt{n}-1}{2\sqrt{n}} \in \mathcal{O}(\sqrt{n})$ (due to X' and the shortened screen Y);

the (minimal-bounding-cuboid) volume of the apparatus lies in $\mathcal{O}(\sqrt{n})$, and is, hence, *exponential* in the size of n.[60]

Let us put this into perspective. In order to factorize an arbitrary, 100-digit number (which is a relatively unambitious aim in, for example, the context of factorizing real-life RSA keys—the 1988 book [132] describes as 'reasonably safe' the use of a 200-digit key), we can expect the depth of the apparatus to be a

[58] We recall from Sect. 2.3.2 *Apparatus* that this interference pattern is influenced by the presence of a *standing wave* along screen Y, the instantiation of which wave, we crucially note, does not affect the time or space complexity (or, for that matter, the precision complexity, which we discuss in Sect. 2.3.7 below) of the system.

[59] Note that, under certain implementations, the sensor with which we identify M-points is required to 'scan' Y in time linear in the length of Y (whereas we suppose that this process—and, hence, stage 3—takes constant time). This length is $\frac{n-1}{2\sqrt{n}} \in \mathcal{O}(\sqrt{n})$, in actual fact rendering the system's time complexity *exponential* in the size $\log n$ of n. In Sect. 2.3.6, however, we modify the system so that this is no longer a concern—see Footnote 61.

[60] We note also that the algorithmic harness mentioned in Sect. 2.3.5 *Time Complexity* imposes a polynomial memory-space requirement, which, though dwarfed by the *exponential* space requirement of the apparatus, is of interest in Sect. 2.3.6 below.

50-digit number of units, whilst its width is three units. The apparatus must be of the order of 10^{49} times as deep as it is wide; it is necessarily either far too deep to be practicably accommodated or far too narrow for slits with sufficiently small spacing feasibly to be manufactured (or, of course, both).

These considerations render the system unsuitable as a practical solution to the general problem of factorization. We consider now a subproblem, restriction to which greatly mitigates the problems discussed here.

2.3.6 RSA Factorization

We see above that, by identifying an M-point $(0, h)$ on Y, we are able to find factors $p := \sqrt{n}\left(\sqrt{h^2+1}+h\right)$ and $\frac{n}{p}$ of n. The problem is that there is no a priori, n-independent upper bound for h: factors of n are found from values of h as large as $\frac{n-1}{2\sqrt{n}}$, which clearly tends to ∞ as n does. We shall never have made screen Y large enough to cater for all n.

We note, however, that, as h increases, the corresponding factors p and $\frac{n}{p}$ of n grow further apart. In the case where n is a square, there is an M-point at $(0, 0)$, corresponding to the factorization $n = \sqrt{n}\sqrt{n}$: small values of h (in this example, zero) give close (in this example, equal) pairs of factors. At the other extreme, for any natural number n, there is by Prop. 11 an M-point at $\left(0, \frac{n-1}{2\sqrt{n}}\right)$, which corresponds to the factorization $n = 1.n$: large h give greatly differing pairs of factors. So as to quantify this relation, we have the following.

Proposition 14. The factors $p := \sqrt{n}\left(\sqrt{h^2+1}+h\right)$ and $q := \frac{n}{p}$ of n corresponding to an M-point $(0, h)$ on Y differ by $2\sqrt{n}h$.

Proof. We have that $q = \sqrt{n}\left(\sqrt{h^2+1}-h\right)$, for then, as in the proof of Prop. 12, $pq = n$. So

$$\begin{aligned} p - q &= \sqrt{n}\left(\sqrt{h^2+1}+h\right) - \sqrt{n}\left(\sqrt{h^2+1}-h\right) \\ &= \sqrt{n}\left(\sqrt{h^2+1}+h-\sqrt{h^2+1}+h\right) \\ &= 2\sqrt{n}h \ , \end{aligned}$$

as required. □

Suppose now that we modify the system so that the size of the screen Y is bounded; specifically, suppose that Y no longer occupies the line segment $\{0\} \times \left[0, \frac{n-1}{2\sqrt{n}}\right]$, but rather the line segment $\{0\} \times [0, l]$ for some fixed (and, crucially, n-independent), positive real number l. Recalling the discussion of Sect. 2.3.5 *Space Complexity*, then, this gives the system *polynomial* space complexity—we have constant-volume apparatus (contained within a cuboid of volume $3\left(l+\frac{3}{2}\right)\epsilon$ for some negligible, z-axis height ϵ) with a polynomial-memory harness, as discussed in Footnote 60.[61]

For sufficiently large values of n (specifically, those with $\frac{n-1}{2\sqrt{n}} > l$; i.e., $n > 2l\left(l+\sqrt{l^2+1}\right)+1$), we have that Y is no longer large enough to accommodate all M-points corresponding to factors of n: those pairs of factors corresponding to M-points $(0, h)$ with $h > l$ are overlooked. However, we have the following.

[61] The system's time complexity is, further, rendered *polynomial* under all implementations of the sensor on Y (recall Footnote 59).

Proposition 15. If a pair (p, q) of factors of n (with $pq = n$ and $p \geq q$) satisfies $p \leq mq$, where $m = 2l + 1$, then these factors are not overlooked.

Proof. Required is that the y-coordinate h of the M-point corresponding to the factor pair (p, q) does not exceed l, for then this M-point falls on the modified (that is, shortened) screen Y.

By Prop. 14, $p - q = 2\sqrt{n}h$, so

$$h = \frac{p-q}{2\sqrt{n}} \ . \tag{2.37}$$

Since, by hypothesis, $p \leq mq$, we have that $p - q \leq (m-1)\,q$; since, again by hypothesis, $pq = n$ and $p \geq q$, we have that $q \leq \frac{n}{q}$, whence $q \leq \sqrt{n}$. Together, these give that

$$p - q \leq (m-1)\sqrt{n} \ . \tag{2.38}$$

Since, by definition, $m = 2l + 1$, we have that

$$l = \frac{m-1}{2} \ . \tag{2.39}$$

Hence,

$$h \overset{(2.37)}{=} \frac{p-q}{2\sqrt{n}} \overset{(2.38)}{\leq} \frac{(m-1)\sqrt{n}}{2\sqrt{n}} = \frac{m-1}{2} \overset{(2.39)}{=} l \ ;$$

$h \leq l$, as required. □

Corollary 2. If $l \geq \frac{1}{2}$, then each factor pair (p, q) (with $pq = n$ and $p \geq q$) satisfying $p \leq 2q$ is found by the modified system.

Proof. If $l \geq \frac{1}{2}$, then $m := 2l + 1 \geq 2$; so $p \leq 2q$ implies that $p \leq mq$, whence Prop. 15 can be invoked. □

In modifying the system so that Y occupies only the line segment $\{0\} \times \left[0, \frac{1}{2}\right]$, we lose the ability to factorize arbitrary natural numbers; by Corollary 2, however, the system can—at least as far as sufficiency of the length of Y is concerned[62]—still factorize those values n of which each factor p no less than $\sqrt{n}$ (but strictly less than n[Footnote 63]) satisfies $p \leq \frac{2n}{p}$. Further, we note that, for a public key $n = pq$ (with p and q prime and $q \leq p$) of the RSA cryptographic system, this situation, in which $p \leq 2q$[Footnote 64], is common (the intuition here is that, if prime factors p and q are too far apart—if q is particularly small and p particularly large—, then the key $n = pq$ is susceptible to factorization via brute force).

Having noted an impracticality in using the system to factorize arbitrary natural numbers (namely, that, as the input value grows, the required ratio of the depth of the apparatus to its breadth increases exponentially, thus precluding physical implementation), we have nonetheless identified a subproblem—factorizing RSA keys—that this impracticality does not hinder.[65]

[62] This, we see below, offers no guarantee of a practicable system. Chief among other potential complexity bottlenecks are issues of *precision*.

[63] We excuse the system for omitting to demonstrate that $n = 1.n$.

[64] Were we even to weaken this condition from $q \leq p \leq 2q$ to $q \leq p \leq 10q$, say—a very conservative requirement of RSA keys—, then the system would, by Prop. 15 and provided that Y spans $\{0\} \times \left[0, \frac{9}{2}\right]$, still be able to factorize n (again, at least as far as Y's length is concerned). This proviso causes no difficulty in physical implementation.

[65] There is, furthermore, a sense in which this subproblem captures the 'hardest' instances

2.3.7 Precision Complexity

Having restricted factorization to RSA keys, we have a system with polynomial space and time complexity. However, the system suffers from its *precision complexity* (which is formalized in the following chapter), as we now informally explain.

We recall from Sect. 2.3.2 *Input to the System* the interval I_n, to an element of which the wavelength of S must be set in order successfully to factorize n; we recall from (2.32), in particular, that the size of the interval satisfies $\frac{1}{4n\sqrt{n}} < |I_n| < \frac{1}{4\left(n-\frac{1}{2}\right)\sqrt{n+\frac{1}{2}}}$. Since $\frac{1}{4\left(n-\frac{1}{2}\right)\sqrt{n+\frac{1}{2}}} < \frac{1}{4\left(n-\frac{1}{2}\right)\sqrt{n}}$, then, we have that $\frac{1}{4n\sqrt{n}} < |I_n| < \frac{1}{4\left(n-\frac{1}{2}\right)\sqrt{n}}$, and these upper and lower bounds for $|I_n|$ differ by $\frac{1}{4\left(n-\frac{1}{2}\right)\sqrt{n}} - \frac{1}{4n\sqrt{n}} = \frac{n-\left(n-\frac{1}{2}\right)}{4n\left(n-\frac{1}{2}\right)\sqrt{n}} = \frac{1}{8n\left(n-\frac{1}{2}\right)\sqrt{n}}$, which tends to zero as n tends to infinity. Therefore (by the algebra of limits), $|I_n|$ tends to $\frac{1}{4n\sqrt{n}}$ as n tends to infinity; crucially, *the size of I_n shrinks exponentially with the size of n.*

Containing the error (i.e., the discrepancy between the intended and actual wavelengths of radiation from S) to this extent therefore requires of the user a precision (whilst setting this wavelength) *exponential* in the size of n.

Rather than a system that efficiently solves a problem, we have yet another motivating example[66] for extension of complexity theory beyond the essentially Turing-machine. By introducing notions of complexity that, for certain unconventional computers, cater better than those of the traditional theory, the present work represents an attempt at such extension.

(We suggest that the lack of such extension prior to the present project is perhaps because a vast majority of practical computation conforms to a Turing-machine-like model (specifically, real-world computers are typically digital, running programs that implement algorithms), and perhaps also because of an overestimation of the ambit of the Church-Turing thesis (which ambit, we suggest, falls within the realm of computability rather than complexity); consequently, resource is, traditionally, virtually always taken to be a property—usually run-time, but sometimes space or another measure satisfying Blum's axioms—of an algorithm, or, equivalently, a Turing machine, random-access machine or similar.)

Certain points from the preceding paragraphs are worth reiterating and elaborating. Although our factorization system has polynomial space and time complexity, this is not to say that it is a 'good', practicable system. However, the reason for which it is not a good system—namely, its exponential precision complexity—does not form part of a standard, Turing-machine-type complexity analysis; performing a standard analysis, considering only standard resources (run-time, space, etc., but not precision), evaluates overly generously (as polynomial) the system's (actually exponential) cost. We discuss the system

---

of (general) factorization: traditional, general-factorization algorithms typically require the greatest run-time when their input is of the form pq, where p and q are primes of the same approximate size. Also, it is conjectured in [112] that any general method whereby the RSA system is broken—such as an efficient method for factorizing RSA keys—would yield an efficient method for factorizing arbitrary natural numbers, implying the equivalence of the RSA- and general-factorization problems.

[66] We add this example to the famous soap-bubble method [97] for finding minimum-length spanning networks (possibly with additional vertices) connecting given vertices, the DNA-computing technique [3] for tackling the directed Hamiltonian path problem, etc.

here, therefore, not because it factorizes efficiently (it certainly does not[67]), but because it demonstrates the need to consider more broadly the complexity of systems and the cost of computations;

complexity analyses—especially of non-Turing computers—should consider not only algorithmic resources (time, space, etc.), *but also non-algorithmic resources* (precision, energy, etc.).

2.3.8 Summary

We summarize now Sect. 2.3.

In Sect. 2.3.1, we suggest that a possible cause for the prohibitive precision complexity of the system of Sect. 2.2 is its 'handmade' nature, and consider an alternative implementation of the same basic idea whereby physical phenomena (from which result naturally occurring cones) are exploited in order to lessen the need for precise manufacturing (see also Sect. 3.2.1 *Manufacturing Costs*). This motivates an improved factorization system.

In Sect. 2.3.2, we define the system, detailing its apparatus and input/output processes. We go on, in Sect. 2.3.3, to show that these processes result in the correct factorization of the input value.

In Sect. 2.3.4, we discuss and resolve some, though not all, of the practical difficulties with the system's implementation; specifically, we address and solve the problem of the apparatus's being infinite in spatial extent.

In Sect. 2.3.5, we note the system's favourable (polynomial) time complexity, but that its (exponential) space complexity impedes practical implementation. We resolve this difficulty in Sect. 2.3.6 by restricting the system's apparatus, though this restricts accordingly the problem solved by the system: while a practically implementable and efficient system for factorizing arbitrary natural numbers seems precluded by many concerns, we present a polynomial-time, polynomial-space system for factorizing a certain type of natural number—which type includes RSA keys—that is not subject to some of these concerns.

In Sect. 2.3.7, however, we note that the system's ostensibly impressive polynomial time- and space-complexity functions are testament not to the system's efficiency, but to the inadequacy of traditional, Turing-machine-motivated complexity theory for capturing/accommodating certain unconventional computers' complexity. The true complexity of the system, which is exponential by virtue of the precision required during its use, is overlooked by traditional complexity analyses.

This observation motivates the present project's attempt to introduce an approach to complexity theory that caters more broadly than for Turing-machine-style computers alone. Specifically, the aspect of this project motivated by the factorization systems of Sects. 2.2 and 2.3 is the consideration of non-standard resources (precision being our illustrative example) in the context of analysis of the complexity of non-standard computers.

[67] Neither do we expect any mere modification of the system to achieve polynomial-resource factorization: the issues of precision seem intrinsic and fundamental to the system's derivation. The prohibitive precision complexity is not an obstacle to be overcome; it is an indicator that *non-standard computers warrant non-standard complexity analyses.*

2.4 Conclusion

2.4.1 Summary

The purpose of this chapter is to motivate the present project's introduction of a model-independent approach to complexity theory. The chief such motivation is offered by the thesis that

> *the successful analysis of unconventional computers' complexity entails consideration of unconventional* (i.e., non-time, non-space) *resources*.

We summarize now the observations of the present chapter that lead to this thesis.

In Sect. 2.1, we introduce the problem of *factorization*, noting its exponential time complexity when approached via traditional, Turing-machine methods.

In Sects. 2.2 and 2.3, we describe two unconventional systems that solve the problem of factorization[68]. We investigate the computational complexity of the systems, finding that the conventional (time and space) complexity of each is *polynomial*, but that the informally introduced, unconventional *precision complexity* (formalized below—see Sect. 3.3.3) is *exponential*. This prohibitive complexity behaviour, then, is overlooked by conventional analysis.

(Additionally, the chapter serves a practical purpose by demonstrating the way in which computers may be derived and analyzed in accordance with the model-independent framework described in the present project.)

2.4.2 Discussion

We say no more here than to reiterate that the issue is one of *resource*: a crucial factor that distinguishes conventional and unconventional computation is that different types of resource are consumed in each case; understandably, then, traditional complexity theory fails, due to its restricted interpretation of resource, to cater for some non-traditional computers.

In Chap. 3, we begin (and in Chap. 4 continue) formalization of a model-independent notion of resource.

[68] Properly speaking, the latter system solves a slightly different problem, though one conjectured to be equivalent to that of factorization—see Footnote 65.

Chapter 3

Resource

3.1 Unconventional Resources

3.1.1 The Need Therefor...

Computation—whether performed by a Turing machine, 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/computer have a bearing on its efficiency or lack thereof, and of which values of these parameters and attributes confer this efficiency, is to some extent context-dependent[1], though computational complexity theory offers a widely accepted and highly formalized criterion for efficiency.

Specifically, 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 (where 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* (see Sect. 3.7.1). By construction, then, this limited view of resource is adequate when analyzing the complexity of *standard* computers such as the Turing machine and closely related physical instantiations[2] such as the digital computer. However, as we see in Chap. 2—see *constructively*, moreover—, consideration of time and space alone can lead to an unrealistically optimistic measurement of the complexity of *non-standard*, *unconventional* computers: it is perfectly possible for a computing system to be rendered impracticable by complexity concerns other than those of time and space; such systems suffer due to prohibitive consumption of some other, non-standard resource.[3] The problem, then, is that unconventional computers may well consume unconventional resources,

[1] For example, portable but non-critical devices may favour space- over time-efficiency, whilst supercomputers striving to process in a timely fashion meteorological data may not.

[2] Adequacy for even these 'closely related' instantiations, however, is debateable—see Sect. 3.6.1 *Actual, Physical Implementations of Turing Machines*.

[3] Explicitly, these non-standard 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.

and complexity should therefore be, though often is not, analyzed in terms of these resources.[4]

Concretely, Chap. 2's systems factorize numbers using time and space growing only *polynomially* in the numbers' size (this is in contrast with the *exponentially* scaling time required by known Turing-machine-style methods for factorizing [41]); however, the *precision* with which the user must manipulate 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—consideration of conventional resources alone leads to an underestimation of the system's complexity, since it is the overlooked, *unconventional* resource of precision (which we formalize in Sect. 3.3) that gives rise to the bars on the systems' practical efficiency, and hence gives rise to the 'true complexity'—see Sect. 3.2.3—of the systems.

Crucially, then,

insightful analysis of the complexity of unconventional computers entails consideration of unconventional resources.

The situation is not beyond repair. On the one hand, traditional complexity theory and the resources (namely, time and space) considered therein are inspired by the Turing-machine model of computation, almost totally to the exclusion of other models; further, these resources cater inadequately for certain non-standard computers (the 'by-construction' adequacy for standard, Turing-machine-like computers notwithstanding). Fortunately, on the other hand, we have that it is possible (and even natural, once the problem has been acknowledged) to define resources that better capture the true complexity of non-Turing computers; this we see from the illustrative case of precision—informally and intuitively in the context of the factorization systems above (see Sects. 2.2 and 2.3) and more formally with our rigorous treatment of precision (see Sect. 3.3)—, and from our discussion below of other unconventional resources (see Sects. 3.5 and 3.6).

The problem, then, is that complexity analysis of unconventional computers is often undertaken with some appropriate resources neglected. Given this formulation, the solution suggests itself: when working with a non-standard computational model, one should consider which resources—both standard (time and space) and non- (precision, the resources discussed in Sects. 3.5 and 3.6, etc.)—are consumed during a computation, and should explicitly measure the complexity of the computation with respect to all of these resources; this gives a more complete picture, and correspondingly more confidence in the understanding, of the computation's complexity.

Aside. We demonstrate here the need for consideration of unconventional resources by first tacitly assuming the need (or at least desire) successfully to analyze the complexity of unconventional computers.[5] A *computer-centric* motivation for this latter (and, hence, indirectly, former) need is a desire to measure the

[4]The obvious corollary is that many physical computing systems, such as the analogue computers of Chap. 2, do not fit meaningfully into the traditional hierarchy of complexity classes, simply because these classes are defined in terms of measures—usually time though sometimes space—that do not accurately reflect the systems' complexity.

[5]It is, of course, only relative to this latter need that we can demonstrate the former: heed of unconventional resources may well be far from necessary if insightful complexity analysis is not amongst our aims.

efficiency of non-standard systems that solve a problem against the benchmark of the efficiency of existing Turing-machine counterparts that solve the same problem—see Sect. 4.1.1 *Turing-Machine Benchmarks*; a *problem-centric* motivation is that with the ability successfully to analyze unconventional computers' complexity comes an improved understanding of the complexity of mathematical problems themselves—see Sect. 4.1.1 *Complexity: Problems versus Solution Methods*.

More generally, there has long been and is today an active community working with non-Turing forms of computer[6], using such techniques as

- mechanical means whereby differential/integral equations are solved (e.g., the Differential Analyzer—see [43] and Example 3);
- the formation of soap bubbles between parallel plates, as used to find minimal-length spanning networks (possibly with additional vertices) connecting given vertices—see [97];
- DNA-computing techniques that can tackle the directed Hamiltonian path problem—see [3] and Sect. 3.6.1 *Chemical/DNA Computers*—, amongst other graph-theoretic/combinatorial problems;
- quantum-computing methods, both standard (circuit-model [99]) and non-standard (adiabatic [87,88], measurement-based [109], continuous-variable [39], etc.)—see Sect. 3.6.2;
- optical means offering novel approaches to the Travelling Salesman and other problems—see [72, 133, 134] and Sect. 3.6.1 *Optical Computers*;
- and so on.

Despite the dominance of digital computers, then, non-Turing systems are of great importance (and, moreover, increasing importance, thanks largely to quantum computation).

It is given the need for insightful analyses of non-standard computers (and given Sect. 3.1.1's dashing of the hope that the traditional tools—especially the traditional resources—of complexity theory are adequate for analyzing unconventional computers), then, that one must consider new resources.

This need for consideration of unconventional resources corroborates in a specific way Penrose's claims [105] that

> "the questions of practicality of algorithms are being only barely touched by complexity theory as it stands today"

and that

> "complexity theory for *actual physical objects* could perhaps be different in significant ways from that which we have just been discussing [i.e., from that for Turing machines]".

Further, we see presented in [119] a framework in which can be made complexity analyses (to an extent such that these analyses are meaningfully comparable with those of discrete systems) of *dissipative flows*, which model continuous

[6] We note also the growing recognition (see, e.g., [2, 78]) of such computers' importance.

physical systems of which the evolution can be viewed as computation (where the output is—or, at least, is encoded in—the attractor of the dynamics). The authors of [119] conjecture that a problem's tractability (in the sense of its having polynomial time complexity) via Turing machine is equivalent to its tractability (in a sense defined in [119]) via dissipative-flow computation. Notably for our purposes, [119] acknowledges the inadequacy of traditional complexity theory:

> "Although continuous time systems are widespread in experimental realizations, no theory exists for their algorithmic analysis. The standard theory of computation and computational complexity ... deals with computation in discrete time and in a discrete configuration space, and is inadequate for the description of such systems."

Despite this acknowledgement, however, the consideration in [119] of resources consumed during computation via the above-mentioned continuous-time systems is restricted to run-time.

Aside. Whereas [119] discusses the continuity of *time* in the context of computation via dissipative systems, an analogous treatment of *space* leads to the rudiments of something like our notion of precision. There is in the present work and [119], further, a common use of continuous-to-discrete 'correction' via rounding or similar:

> "The evolution of a [dissipative dynamical system] reaches an attractor only in the infinite time limit. Therefore for any finite time we can compute it only to some finite precision. This is sufficient since for combinatorial problems with integer or rational inputs, the set of fixed points (the possible solutions) will be distributed on a grid of some finite precision. A computation will be halted when the attractor is computed with enough precision to infer a solution to the associated problem by rounding to the nearest grid point" [119].

Compare this with our notion of *interpretation* (see Definition 6).

Finally, we find in [81] a description of an implementation of a DNA computer that solves the shortest path problem. We read that,

> "[e]ven though the shortest path problem is belonging to the class P, i.e., it is not hard to solve this problem, it is worth to be solved by DNA computing because numerical evaluations are required during the computation."

The suggestion seems to be that, although the *time* cost of solving the problem via Turing machine is reasonable, it may be desirable to minimize not run-time but rather the number of *numerical evaluations* performed during the computation; more abstractly,

> *we may be interested in resources other than time (and space)*,

a thesis with which we strongly agree. (The fact that these troublesome numerical evaluations presumably impose only a time cost means that consideration of both time and numerical evaluations boils down to consideration of time alone, but this is not the point; though the role of resource seems confused in the above quotation, one imagines that this is merely because the issue has not

been explicitly considered by the authors of [81]—that they challenge at all the supposition that 'time is enough' is commendable, and that they lack a framework in which to formalize resource issues (which formalization we attempt in the present work) is no cause for criticism, for such is not their aim.)

We must make clear that, whilst we argue for consideration of unconventional resources, we certainly do not argue for consideration of such resources *alone*. Of course, one should still consider as part of the complexity analysis of a given, unconventional computer the *standard* resources of time and space; however, especially if these suggest unexpectedly low complexity (as, for example, do the polynomial time- and space-complexity functions of the factorization systems of Chap. 2), *non-standard* resources should also be considered.

3.1.2 . . . But the Neglect Thereof

Despite the need—which is hopefully clear given the discussion of Sect. 3.1.1—for consideration of unconventional computational resources, there is, nonetheless, a widespread tendency to overlook such resources during complexity analyses. We justify here this assertion[7], and go on in Sect. 3.1.2 *Explanation* to offer possible explanations for the neglect discussed here.

Regardless of its causes, however, this misapprehension is widely evident; we list some manifestations.

- According to its scope [66], the *International Workshop on Foundational and Practical Aspects of Resource Analysis*

  > "serves as a forum for presenting original research results that are relevant to the analysis of *resource (time, space)* consumption by computer programs" (the emphasis is ours; the restricted view of resource is not).

  The stipulation that considered systems be restricted to "computer programs" makes suitable the exclusive consideration of time and space (see Sect. 3.7.1), but this restriction seems not to be in keeping with the workshop's claimed consideration of foundational issues concerning resource.

- In [30], we read that

  > "[t]he cost of computing a function by a machine reflects the number of fundamental operations performed from input to output".

  This, we suggest, results in a complexity measure essentially the same as *run-time*—see Sect. 3.6.2 *Circuit-Model Quantum Computers* for discussion of a similar resource in the context of quantum computation. A more explicit demonstration of the neglect of non-standard resources is the claim in [30] that

  > "[p]olynomial time is meant to formalize the notion of tractability";

[7] See also Sect. 3.7.1 for a related, almost converse assertion: that, *in the Turing-machine case*, time and space are the only resources reasonably to be considered.

not even space, then, is entertained as a valid resource. This overemphasis on time is particularly dangerous in [30] since the real-number manipulations considered therein can reasonably be expected to imbue any practical implementations of the corresponding systems with non-trivial precision complexity.

- Reference [38] has a similar approach to [30], with accordingly similar neglect of non-standard resources.

- Reference [42] discusses a soap-bubble method for finding minimal-length Steiner trees connecting given vertices (see, for example, Chap. VII, §11, Subsection 4 of [52]), though this discussion neglects to consider non-standard (in fact, non-time) resources *when such are particularly relevant.*[8]

- In its consideration of complexity, [40] limits itself to the resource of time. This leads to the authors' making such claims as

  > "[Shor's polynomial-time factorization system] would probably violate the [extended Church-Turing thesis—see Sect. 1.2.1], since factoring is believed to require superpolynomial time on a classical computer";

  however, we have in Chap. 2 exhibited a polynomial-time, 'classical' (i.e., non-quantum) factorization system: whereas factorization seems to require super-polynomial *resource*, the requirement for *time* in particular need not be so prohibitive.

- And so forth.

There are, of course, exceptions in which unconventional complexity resources are considered. We recall, for example, [10]; whilst not treating precision as a resource in its own right as we do here, the work effectively employs an equivalent formulation[9] by acknowledging that an increase in precision *imposes a cost* in terms of other resources (specifically time).

The particular example treated in [10] is that of a method for determining which of two objects is the more massive. By elastically and one-dimensionally colliding one (moving) object, A, into another (stationary) object, B, on a frictionless surface, the transfer of momentum is such that,

- if A is the more massive, then it continues (albeit more slowly) in its original direction as if through B—this situation is identified by A's passing an anterior sensor;

- if A is the less massive, then it recoils from B so as to travel oppositely to its original direction—this situation is identified by A's passing a posterior sensor; and

[8] This oversight leads the authors of [42] to conclude that $\mathsf{P} = \mathsf{NP}$; the most flawed step of their argument is that, since the (NP-complete) problem of finding minimal Steiner trees can be solved in polynomial time by an analogue computer, there must exist a *similarly polynomial-time* Turing machine that solves the same problem. However, there is no reason to think that the conversion to Turing machine can be executed without, for example, having to introduce exponential time complexity so as to accommodate an exponential consumption by the pre-conversion, analogue system of an unconventional resource such as precision.

[9] Equivalent, that is, once certain trade-offs are established—cf. Sect. 3.8.1 *Precision and Time.*

- if A and B have exactly equal mass, then A remains at the point of collision, never to reach either sensor.[10]

See Fig. 3.1.

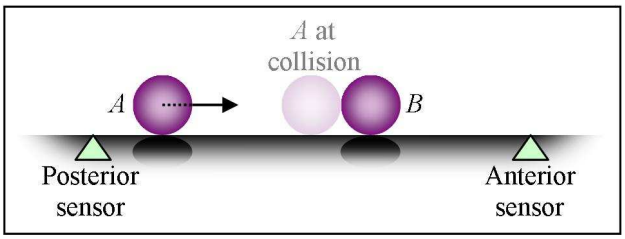

Figure 3.1: The experiment of Tucker et al. [10], which determines which of A and B is the more massive.

The claim (the justification of which we defer to [10]) of interest here is that the *time* for which a user of the system need wait in order to guarantee that A reach one or other sensor increases exponentially with the required *precision* (which is related reciprocally to the smallest difference in mass that we wish to be able to distinguish). Whilst precision is not modelled as an atomic resource, then, it is certainly considered (via its relationship to standard resources).

Aside. We note that the *energy* of the system can be increased in order to reduce the time for which the user need wait for an answer: by doubling the initial velocity of A, for example, the waiting time is halved. This hints at the richness of the choice of unconventional resources, and at the existence of trade-offs between resources—see Sect. 3.8.1.

We observe another step very much in the right direction in [128]:

> "The amount of tape used by a Turing machine computation can be no larger than the number of time steps, and *it uses no resources other than time and* [*space*]. Therefore, a problem [in P] is also guaranteed to use no more than a polynomial amount of resources on a Turing machine. On the other hand, an analog computer can conceivably operate successfully in polynomial *time* but require an exponential amount of *some other resource*, such as torque or instantaneous current. We will therefore want to insist that a "fast" and well behaved analog computation use *total resources* polynomial in the input description" (our emphases).

Explanation.

We offer now some potential explanations for this neglect (above exceptions aside).

[10]However, B *will* in this case pass a sensor (namely, the anterior), having adopted after the collision the velocity and direction of A before. This suggests a method of ascertaining comparative mass in reasonable time even when the masses are equal or nearly so; the crucial dependency of time is then upon the energy available to impart to A at the start of the experiment rather than upon the mass difference between A and B. We defer full exploration of this idea to future work.

- The most obvious is that a vast majority of practical computation conforms to a *Turing-machine-like model* (specifically, real-world computations are typically performed by digital computers that run programs that implement algorithms that can be formalized as Turing machines, random-access machines or similar), and, hence, warrants consideration in its complexity analysis of only time and space—see Sect. 3.7.1. Perhaps understandable, then, is the misapprehension that consideration of time and space alone suffices *even on those rare occasions when the computational model is not Turing-machine-like*.

- Further, there may be an overestimation of the ambit of the Church-Turing thesis (which ambit, we suggest, falls within the realm of computability rather than complexity—see Sect. 1.2.1). Supposing even the *extended* thesis, the belief is still mistaken that all computing paradigms, conventional and unconventional, are in some sense of equivalent computability and complexity *on a resource-by-resource basis*, and, hence, that complexity analysis à la any arbitrarily chosen model (the natural choice being the Turing machine with its well-established techniques and tools of complexity analysis) is adequate in all cases.

- Complexity theoreticians may understandably wish to decide upon and work exclusively with concrete notions (such as 'Turing machine' instead of 'computer', or 'time/space' instead of 'arbitrary resource'), rather than retaining model-generality at the cost of having less specific definitions from which fewer general theorems follow (see also Footnote 13).

Because of these and other considerations, the temptation may be to analyze the complexity of an arbitrary computer as one would a Turing machine;

the temptation may be, in particular, to neglect non-standard (i.e., non-time, non-space) computational resources.

This oversight is seldom explicitly considered, let alone viewed as cause for concern; indeed, the field of computational complexity theory, even restricted to Turing-machine-style resources, is extremely successful, not only theoretically, but also practically, offering excellent guidance, for example, on the allocation of computational effort.[11]

Lack of Unification.

We discuss now a problem related to the neglect of unconventional resources, namely, the lack of unification in the way in which such resources are defined/treated (on those few occasions when they are considered at all).

It must be emphasized that, Sect. 3.1.2 notwithstanding, there exist (though only in a minority) practitioners of unconventional computing who conduct complexity analyses of their systems with consideration not only of the conventional resources of time and space but also of certain unconventional resources. However, this consideration is not always made to a useful extent[12], and is only ever

[11] Explicitly, we have as an illustrative example of this guidance that NP-completeness is reason enough to call off the search for an efficient algorithm—recall Sect. 1.1.

[12] Often, we reiterate, the true complexity of unconventional computing systems is overlooked—see the discussion of Sect. 3.6.2 *Circuit-Model Quantum Computers* for a timely and important example.

ad hoc, in that the approach to unconventional complexity analysis of (e.g., the inclusion and handling of unconventional resources in) a certain computational model—or even a specific computer within a model—is not intended to be, nor is it, applicable to other models/computers: there is no unification even within individual models, let alone therebetween.

The approach to computational complexity expounded in the present dissertation, with its concepts defined relative to and in terms of (arbitrary) resources in a uniform way, hopefully addresses these problems: in deriving from each resource in a standardized way the corresponding notions of complexity function, complexity class, etc., we set the stage for meaningful comparison between not only instances of a single computational model but also computers conforming to different paradigms.

3.2 Interpreting 'Resource'

Much of this section is based on [23].

3.2.1 Possible Interpretations

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" [111].

In light of this, 'resource' could validly be taken, in the context of computation, to mean not only

1. those 'commodities' consumed/required during computation (i.e., "when necessary" as in the above quotation)—time, space, precision, etc.—,

but also such things, on which we may draw for computational gain or convenience, as

2. the presence of non-determinism (which, supposing the widely held conjecture that $\mathsf{P} \neq \mathsf{NP}$, affords a strict increase in computational efficiency),
3. the consultation of oracles,
4. the costs incurred 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 the thermodynamic costs of the manufacturing process—see [94, 137]),
5. information-theoretic resources,
6. etc.

One hopes that several of these diverse interpretations of 'resource' can be reformulated as instances of a single, more general interpretation (see Sect. 3.2.2 for discussion of this phenomenon), but that this interpretation is nonetheless sufficiently specific to form the foundation of an insightful complexity framework

in which meaningful results can be derived.[13] We adopt a view of 'resource' that, prima facie, seems much more specific than is suggested by the word's common usage (in the form of its dictionary definition) and by the variety amongst points 1 – 6 above; we see below, however, that our definitions are able to accommodate many of these points.

Specifically, we are concerned primarily with the '*commodity resources*' of point 1 (as is implicit in the informal description of Sect. 1.2.1 *Resource*). These are computational resources consumed or required by a computer during execution of a computation (as opposed[14], for example, to resources used during a computer's manufacture, and to information-theoretic resources such as communication channels). As already mentioned, commodity resources include, but are by no means limited to, time and space; the notion therefore generalizes its counterpart in standard complexity theory.

Before focusing on commodity resources, we briefly discuss some of the alternative interpretations alluded to above.

Manufacturing Costs.

One may view as a resource the costs incurred in *constructing* (as opposed to running) a computer. These costs may, for example, include those of manufacturing the system's *physical structure*, or of producing an *entangled state*[15] to be used in a quantum computation (e.g., a *cluster state* from which measurement-based quantum computation proceeds—see [109]).

An example of a physical-structure manufacturing cost notable for its enormity comes from *relativistic computing* (see Sect. 3.6.1 *Relativistic Computers*), some instances of which make use of wormholes, black holes, etc., which, if not already present, may need to be constructed (such manufacturing costs are suggestive less of hypercomputation and more of hypercompartmentation, so far beyond the pale are they...).

In the world of practical computation, such costs are typically encountered in a *financial* form (e.g., the price of a digital computer), though may instead be

[13] This is, of course, an instance of the age-old balancing act of mathematical definition. If a notion is too broad, then few interesting theorems hold in generality (the larger the class X being defined, the fewer interesting properties P satisfying $(\forall x \in X)\ P(x)$ there are); if the notion is too narrow, then it is unlikely to be applicable in an arbitrarily chosen situation of interest, and resultant theorems potentially lack insight in that their predictions may be artefacts of the overly restrictive definition rather than being inherent truths.

[14] An attempt at a fully rigorous definition of 'commodity resource' that, in particular, distinguishes the notion from the other categories of resource listed above would be artificial: see Sect. 3.2.2.

[15] Reference [68] sets out several principles of which the intention is to constrain and specify what is meant by 'computing machine' (we begin similar axiomatization of 'resource' in Sects. 3.4 and 4.5). Of particular interest is Principle IV, which stipulates that causation be local: justified by

> "the finite velocity of propagation of effects and signals [and the fact that] contemporary physics rejects the possibility of instantaneous action at a distance",

the principle demands that, for each region of a machine, the current state of only a bounded neighbourhood of the region influence the next state of the region. Notably, this principle precludes disparate, entangled states (e.g., Einstein-Podolsky-Rosen pairs; see [61] for discussion—indeed, introduction—of the associated paradox), which may in turn preclude otherwise-possible computations and certainly precludes otherwise-(theoretically) possible cryptographic protocols—see [62]. So as to accommodate such, we do not adopt this principle.

thermodynamic, for instance. We recall the notion of *thermodynamic depth*[16], formulated in

- [137] as the loss of information during formation of an object (such, relevantly for present purposes, as a computer), and in
- [94][17] similarly as the information content of the process of a computing system's creation.

The measure, therefore, falls under the heading of 'manufacturing costs'.

Aside. We mention in passing a concept related to the change in information/entropy during creation of a computer (i.e., related to a computer's thermodynamic depth), namely, the change during *resolution* of a probabilistic state or similar. Reference [95] suggests that, intuitively, the complexity of a probabilistic state can be thought of as the work required during the state's resolution. Whereas thermodynamic depth is a manufacturing cost, then, the thermodynamic cost of resolution of a probabilistic state is—assuming that such resolution is a necessary step in producing a computation's output value, and by virtue of its chronological position in the computational process—a commodity resource.

In addition to the costs (be they financial, thermodynamic or other) incurred during *production* of a computer, one may consider also those incurred during its *design*; for our purposes, such are included in the category of manufacturing costs, being incurred as they are before the computation itself commences. Apart from obvious instances of these design costs, such as the man-hours required to design a digital computer[18], this type of cost also includes *evolution*: viewing (e.g., human) brains as computing systems—and why should we not?—, one may reasonably consider as an associated cost the time taken to evolve the brain.[19]

Note that the evolutionary costs of a human-brain computer (not least the evolution time) dwarf the production (gestation) and—for even vaguely tractable problems—computing (thinking) times. Nonetheless, this evolution imposes a one-off cost, a $\mathcal{O}(1)$ (albeit huge) resource expenditure—see the following paragraph. Our choosing to study commodity resources rather than manufacturing costs, then, is similar to standard complexity theory's simplification 'polynomial is tractable', even when the polynomial's coefficients/degree are impracticably large. Ultimately, our aim is to study the complexity of (especially non-Turing) computational processes, not the complexity of the computers' coming into being; our choice of commodity resource is in keeping with this.

Ultimately, for a computing device to have a serious, *complexity-theoretic* claim to solution of a problem (and not merely solution of a proper subset of

[16]This is to be contrasted with the thermodynamic cost, also discussed in [137], incurred *during* a computation—see Sect. 3.5.1 *Thermodynamic Cost*.

[17]Incidentally, in agreement with our thesis that unconventional computers warrant unconventional resources, we read in [94] 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".

[18]Depending upon how we define 'design', we may wish also to include the man-hours required to design the computer's *predecessors* from which its design derives.

[19]See also Sect. 3.6.1 *Chemical/DNA Computers*.

the problem's instances—say those of a specific size), the same device should be capable of processing any valid input value of the problem; more difficult (usually larger) instances may require more ('commodity') resource (such as run-time or memory space) during computation, but the *initial construction* of the computer is taken by the complexity theorist as read, and the details of this construction, being independent of input value, impose only a *constant* cost, which is in particular ignored by asymptotic notation.

Therefore, we, as complexity theorists, are interested more in those resources consumed during computation than those that constitute manufacturing costs.

As a final point in our discussion of manufacturing costs, we stress that, even when discussing, say, the computational paradigm of the physically implemented digital computer,

memory space is not a manufacturing cost, but rather a commodity resource.

If the amount of physical memory required depends upon the specific instance of a problem under consideration and is unbounded—i.e., given any finite allocation of space, there exists an instance of the problem that, due to lack of space, cannot be solved—, then we cannot a priori (and, in particular, before knowing for which input values to cater) guarantee sufficient memory; it may be that further memory space is required *as the computation proceeds*, hence the thesis above.

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 employ *non-determinism*; then, although the same problems are *computable*[20], we do at least enjoy an apparent (and, if $\mathsf{P} \neq \mathsf{NP}$, a genuine) increase in (time) *efficiency*. Similarly, a computer may be augmented by allowing consultation of *oracles*.

Such permissions, which are either a priori allowed by the computational model in question or a priori forbidden (depending on the model), are non-commodity resources on which we may draw in order to aid computation.

Though the unconventional complexity framework that we present in this project does not include as *resources* such features (non-determinism, oracles, etc.), they are nonetheless accommodated—in that we are able to investigate their impact upon the computational complexity of problems, computational processes, etc.—by *model-independence* of the framework. Though we do not consider non-determinism, for example, to be a *resource*, we are able within the framework meaningfully to compare respectively deterministic and non-deterministic solution methods for a problem, for example.

Regarding the consultation of oracles, we recall from Sect. 14.3 of [103] that, relative to some oracles, $\mathsf{P} = \mathsf{NP}$, and, to others, $\mathsf{P} \neq \mathsf{NP}$. That is, letting a Turing machine that can *consult oracle* A be one that can ask in one time step 'is a in A?', ending accordingly in state q_{yes} or q_{no} so as to steer subsequent computation, and letting C^A be the analogy for such machines of arbitrary time complexity class C, we have that there exist oracles A and B such that

[20]This is because deterministic Turing machines can simulate arbitrary non-deterministic Turing machines (though not, as far as anyone knows or many believe, without increased time and/or space complexity)—see Theorem 2.6 of [103].

$\mathsf{P}^A = \mathsf{NP}^A$ (Theorem 14.4 of [103]) but that $\mathsf{P}^B \neq \mathsf{NP}^B$ (Theorem 14.5 of [103]). Further, we recall from [17] that, for a randomly chosen oracle A, $\mathsf{P}^A \neq \mathsf{NP}^A$ with probability 1.

These theorems render natural such questions as 'do the abilities of non-standard (analogue, quantum, etc.) computational paradigms correspond (in terms of either computability or complexity) to consultation by a Turing machine of some oracle?' (cf. the approach of [9,10], for example) and 'do the theorems hold when Turing machines are replaced by other (analogue, quantum, arbitrary, etc.) computers?' We defer investigation of these issues to future work.

Features of Enveloping Physical Laws.

Similarly to the features of the *computational model* (see Sect. 3.2.1 *Features of Computational Models*), we may exploit features of the *physics* describing the model. For example, a quantum computation may rely on the presence of *entanglement*[21], which entails our adopting non-classical physics, whereas, when using other computation models (such as centimetre-scale or larger kinematic computers), it may be convenient to assume that entanglement (and other non-classical phenomena) are *not* present, and that we may safely (given the negligibility of such effects) 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); and, again, they are accommodated (though not as resources) in our framework by virtue of its model-independence (recall Sect. 3.2.1 *Features of Computational Models*).

Information-Theoretic Resources.

Another class of non-commodity resources on which we can draw for computational gain is that of 'information-theoretic' resources. These include (classical and quantum) *communication channels*, *entangled states*[22], etc.

We may, for example, summarize the information-theoretic content of the quantum teleportation protocol [16] 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 (in that they can simulate) resources Y.

Though such information-theoretic entities are not resources in our sense (i.e., are not *commodity* resources), their combination here via inequalities has much in common with trade-offs between commodity resources; see Sect. 3.8.1.

[21] In Sect. 3.2.1 *Manufacturing Costs*, we discuss entanglement in the context of manufacturing costs, and indeed its preparation does impose such; more fundamentally, however, the use of entanglement has as a prerequisite its being physically possible.

[22] We encounter entanglement once again. That it and other resources fall into more than one category is discussed in Sect. 3.2.2

Commodity Resources.

We focus, instead of on any of the above-discussed categories of resource (many of which are, nonetheless, accommodated via model-independence or similar), on *commodity resources*.[23] We introduce and discuss these in Sect. 3.2.4, developing the notion in Sects. 3.4 and 4.5.

3.2.2 Recasting as Different Resource Types

Note that the boundaries between categories of resource (commodity, model-feature, physics-feature, information-theoretic, etc.) are not always clear: there is not always a unique category 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[24]).

However, the crucial point is not whether resources fall neatly into these categories, but that practitioners of unconventional computing consider all resources (whichever categories they may be from) relevant to their systems. It is merely that it is convenient for our purposes to have a specific axiomatization of 'resource'—we use for this purpose our *commodity resource*—; other, non-commodity resources may be catered for in different ways (e.g., via model-independence).

3.2.3 Source of Complexity

Just as we consider in Sect. 3.2 the interpretation of *resource*, one may equally question what is meant by, and whence arises, *complexity*. Though we introduce 'complexity' in a specific, technical sense—see Definition 18—, it is relevant to the present work to consider some alternative, informal interpretations of the word (though remaining within our context of *computational* complexity).

By '(computational) complexity', one may mean

- the amount of work done by a computer;
- the amount of work done by the computer's environment/the universe in which the computation is performed;[25]
- the resource consumed/required during computation;

[23] Because of this focus, a more rigorous description of 'non-commodity resource' than is presented in Sect. 3.2.1 is not necessary; furthermore, such description is rendered elusive by (amongst others) issues described in Sect. 3.2.2.

[24] Though this does not fully divorce non-determinism from the model: the acceptance condition still needs to be suitably chosen (e.g., '*there exists* a choice of non-deterministic bits such that the deterministic version of the machine accepts').

[25] Note that many forms of unconventional computation exploit physical laws so as to defer processing to the universe—see Sect. 6.2.4. Whilst we may not want to class as 'complexity' the laborious processing done by the environment, there is at least much to commend *limiting the physical laws* posited in defining our models.

- the user's difficulty in using the system (one would hope that the user, as opposed to the computer, is expected not to perform non-trivial computation, but rather to wait for (or supply other, non-time resources to) the computer; this stipulation brings us back to the commodity-resource view encapsulated in the previous bullet point);
- the difficulty of the problem itself;[26]
- etc.

Relatedly, the cause of complexity may be

- the very problem that is being solved—if a problem is inherently difficult, then any solution method is necessarily resource-expensive, which expense we may deem to constitute complexity;
- a mismatch between the mathematical model and physical instantiation of a computer—consider the differences (the boundedness of memory and of alphabet size, etc.) between Turing machines and their digital-computer 'implementations' (see the discussion of Sect. 3.6.1 *Actual, Physical Implementations of Turing Machines*);[27]
- difficulties in operating within the limitations—imposed by the enveloping laws of physics or otherwise—of the chosen model (e.g., the constrained ways in which measurements must be taken in quantum computing);[28]
- and so on.

We discuss now these and other points in more detail.

Complexity from the Problem Itself.

It is prima facie possible that certain problems are inherently difficult—that certain problems, no matter which computational paradigm is used for their solution, require significant computational resource. Other problems may be tractable within some computational paradigms but not others; certain problems may have efficient Turing-machine solution, for example, but no efficient solution in another equally expressive model of computation. Yet further problems may be efficiently solved using any (reasonable) model of computation.[29]

[26] This is ultimately what interests the majority of complexity theorists. Unfortunately, whilst the complexity of *mathematical problems* themselves is of natural interest, directly mensurable is the complexity only of *methods that solve the problems*; see Sect. 4.1.1 *Complexity: Problems versus Solution Methods*.

[27] Recall also the discussion in [126], which touches upon the choice of computational models when investigating the complexity of protein folding.

[28] Of course, if practicable implementation of a computer is considered in seriousness, then one presumes that the chosen paradigm, and the laws of physics that come with it, confer some computational advantage, but this advantage may come at a price. For example, one may for computational gain exploit certain quantum-mechanical phenomena (entanglement, superposition, etc.), but will then necessarily be constrained by the 'rules of the game' when it comes to measurement, etc.

[29] It is not clear, beyond an intuitive notion, what 'reasonable' means here. Regardless, the *identity function* should, one presumes, be tractable; we may, further, stipulate that any reasonable model deem tractable other intuitively 'easy' operations such as natural-number *addition*, though this, of course, begs the question.

Prima facie possibilities aside, whether actual problems are inherently difficult (as opposed to complexity being an artefact of the choice of computer or computational model) is discussed in Sect. 6.2.1; in the present section, though, we acknowledge that the problem being solved may potentially be the source of any complexity encountered.

Complexity from Choice of Model.

An alternative source of complexity to that of Sect. 3.2.3 *Complexity from the Problem Itself* occurs when an inappropriate choice of computer or of computational model is made. It is clearly possible to contrive inefficient computers for any computable problem: any computation can be made inefficient simply by adding unnecessary resource consumption; for example, a Turing machine can be modified so as to pause for a number of time steps after each 'useful' time step.

Another example of complexity arising from the choice of computational model is provided by [5]. The difficulty in this example (which, more than a mere increase in complexity, is an outright preclusion of computability) arises essentially from the contrivance that sufficient input data to tackle the problem cannot, due to limitations imposed by the computational model, be read. More specifically, the situation can be thought of as involving a human attempting (and failing) to read many clocks before they change; this failure is a direct consequence of the way in which the reading operation is implemented in the computational model, rather than of any inherent difficulty in the problem itself. That is, the apparent difficulty (in fact, uncomputability) here arises from a mismatch between the choice of computational model and the problem. This is not, we suggest, what complexity theorists[30] should consider complexity: one hopes at the very least for a 'gentleman's agreement' concerning the availability of necessary data, and more generally concerning problems' being approachable.

These, then, are examples of complexity (or even uncomputability) introduced by the *solution method* rather than the *problem being solved*.

Mathematical/Physical Mismatch.

Once a mathematically-modelled computer is physically instantiated, some of the abstract features of the computer may be lost. For example, when one implements a Turing machine as a digital computer, one no longer has unbounded memory or an arbitrarily large alphabet of symbols. Thus may be introduced additional complexity (since the presence of fewer symbols necessitates longer symbol strings[31] and accordingly increased memory space/run-time) or even uncomputability (since a digital computer's finite memory bound imposes a definite limit on input values that can be processed, whereas an abstract Turing machine models as a dependent variable the (unbounded but typically finite) space required to process an arbitrary input value).

This potential source of complexity is not of problematic consequence in the present work since we view as distinct models the Turing machine and

[30] We stress that the author of [5] is not primarily a complexity theorist, and that his work should not, therefore, be judged solely in the context of that field.

[31] Strings taken from an alphabet of a symbols will typically be approximately $\log_a b$ times longer than those from an alphabet of size $b > a$. Cf. the expression of natural numbers in place-notation systems with various bases.

the digital computer (for example): there can be no ambiguity as to whether, say, an unbounded alphabet is permitted. Moreover, we can in any case make meaningful comparisons between a Turing machine and its digital-computer counterpart—see Footnote 7 of Chap. 4.

Aside. Note that the motivation for—and much of the flavour of—the present project is concerned with computation *as physically implemented*, in particular with its sensitivity to input/output imprecision. Not only do we treat (as mentioned in the previous paragraph) actual, implementable computers as distinct from their mathematical abstractions, then, but we also have a natural bias towards the former; whilst we wish our novel approach to computational complexity to agree with the existing approach as it pertains to Turing machines, and whilst we implement our framework of complexity in accordance with this wish, we are primarily interested in the new insight that our approach offers in non-Turing cases.

Physically-Imposed Difficulty.

We may view as complexity the difficulty encountered when, for example, taking measurements from a classical, quantum or other computing system. This, we suggest in passing, has an equivalent formulation in our 'commodity resource' view: we may model this difficulty as a requirement for precision, or for time, or for some other resource.

3.2.4 Commodity Resources

We turn now to our chosen interpretation of 'resource': what we have alluded to already as the *commodity resource*.

Justification of Choice.

Our interpreting 'resource' in the commodity sense (rather than in accordance with any of the alternatives—as features of the computational model/laws of physics, manufacturing costs, information-theoretic resources, etc.—discussed in Sect. 3.2.1) is, we suggest, a natural choice.

As we note above, many non-commodity resources (we have in mind, in particular, features of computational models and of enveloping physical laws) are accommodated by the model-independence of our approach to complexity theory, and so may still be investigated within the framework of this project, despite their not being expressed as resources.

Additionally, this choice follows relatively closely the development of existing, Turing-machine complexity theory—wherein non-commodity resources such as the computational use of non-determinism are accounted for in the classes themselves (consider the difference between the respective definitions of P and NP, for example)—, which we wish to retain as a special case of our model-independent theory.

Recall also the fine balance between a definition's being too general and too specific (see Footnote 13). The notion of commodity resource (which we describe formally below—see Sects. 3.4 and 4.5) is sufficiently general to capture many relevant features of the computational complexity of unconventional computers (with our framework accommodating otherwise than as resources many other

features of interest, moreover), whilst being sufficiently specific that much can be inferred about these resources.

3.3 Precision —an Illustrative Unconventional Resource

Before formalizing the generic notion of (commodity) resource, we return to our illustrative example thereof: *precision*.[32]

We see in Chap. 2 that, much as a standard computer may require that a user wait for a certain *run-time* to elapse before an output value is returned, or that a certain *memory space* be available in order to run (which run-time and memory space will typically increase with the size of the system's input), a non-standard computer may require of the user a certain *precision* with which (error-prone) input parameters must be manipulated and (error-prone) output parameters measured (which precision will typically increase with the size of the input).

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 of the system, 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 or the *difficulty to the user* in performing the computation (cf. Sect. 3.6.2 *Circuit-Model Quantum Computers*)—than resources such as precision (which are not applicable to standard computers—see Theorem 1).

3.3.1 Motivation

In the present project, our chief motivation for considering the resource of precision in a complexity-theoretic context is given by the analogue factorization systems of Chap. 2. We add to this the following brief comments.

Suppose that one encodes a Turing-undecidable set U (the Halting set of Sect. 1.2.1 *Halting Problem*, for example) of natural numbers[33] as a real number x_U between 0 and 1, such that the $(i+1)$st binary place of x_U is 1 if i is in the set U and 0 otherwise[34]; that is, $x_U = \sum_{i \in U} 2^{-i-1}$. Suppose further that an object can be positioned—with perfect precision—so as to have spatial coordinate x_U. Then infinite-precision measurement results in super-Turing computational ability (since the undecidable problem '$\in U$?'—e.g., the Halting problem—may be solved by measuring the appropriate bit of the object's

[32] It is high time, also, that we explain the subtitle, "*From Patience to Precision and Beyond*", of this dissertation; it describes the progression, made chiefly in this chapter, from the traditional-complexity mindset and according consideration of little more than *time* complexity, via the unconventional-computing perspective and corresponding heed of *precision* complexity, to the full model- and resource-independence of the present project's framework of complexity.

[33] Note that, in this and subsequent chapters, zero is deemed natural: $\mathbb{N} := \{0, 1, 2, \ldots\}$.

[34] This encoding (as a function of undecidable sets) is injective. That $0.b_1b_2b_3\ldots b_k0\dot{1} = 0.b_1b_2b_3\ldots b_k1\dot{0}$ (for bits b_i) is not an issue: since U is undecidable, its encoding x_U is irrational (for else the encoding's binary expansion would either terminate after finitely many places or become periodic, whence U would be decidable via finite look-up table), and, in particular, has a binary expansion ending in neither infinitely many 0s nor infinitely many 1s.

position).[35] To be clear, we are suggesting neither that infinite-precision positioning/measurement is possible in the real world, nor, therefore, that it confers a practicably exploitable computational advantage; we are suggesting, instead, that precision needs to be considered and accounted for in order to achieve correct analysis of certain computational systems.

It is for these and other reasons that it is desirable to include precision in complexity analyses.

3.3.2 Examples

Though polynomial-time, polynomial-space factorization is particularly evocative because of the problem's notorious difficulty, the systems of Chap. 2 are relatively convoluted, which, in particular, clouds somewhat the issue of precision. We present now some systems that solve simple problems (which, indeed, can be solved efficiently by Turing machine) in simple ways, thus making more clearly evident the role of precision in these systems' computations.

We analyze the following systems' precision complexity before formally defining (in Sect. 3.3.3) the notion; this, we hope, offers a more concrete understanding of precision once its definition is reached.

Addition.

Suppose that one wishes to find the sum of two given natural numbers n_1 and n_2. This addition can be performed by (naively but illustratively) simplistic physical means[36] by cutting two pieces p_1 and p_2 of some material with respective lengths n_1 and n_2 units, placing the pieces end-to-end, and measuring their combined length.

We consider now the accuracy with which one needs to cut p_1 and p_2 and to measure the combined length in order to guarantee arrival at the correct result $N := n_1 + n_2$. To this end, we model the various errors arising in the system and consider a rudimentary form of error correction via interpretation of non-integer combined lengths. We consider first an additive, and secondly a multiplicative, model of error; which is the more appropriate is determined by the exact details of implementation of the physical process, which details are of less interest in the present context than is the resultant dependency on precision in either case.

Additive Error. It may be the case that the error introduced during the cutting of p_1 and p_2 is best modelled *additively*, in that (for $j \in \{1, 2\}$) addend n_j is represented by p_j's length l_j, which is known only to lie in the interval $[n_j - \epsilon_i, n_j + \epsilon_i]$ for some real *input error* term $\epsilon_i \geq 0$. This occurs, for example,

[35] Of course, knowledge of the mapping $U \mapsto x_U$ from undecidable sets into the interval $(0, 1)$ does not imply sufficient knowledge to set a position to x_U (even given infinite precision): that x_U *exists* does not mean that its value is *known* (in fact, in an exclusively Turing world, its value *cannot* be known). We focus, however, on the specifically precision-related issues present in this hypothetical experiment.

[36] Of course, the computation can equally be performed in linear time and space via a Turing-machine implementation of column-by-column, 'schoolbook' addition. Our aim in taking a physical-computation approach is not to beat (in terms of efficiency or similar) the Turing benchmark, but to highlight the issue of precision in a simple instance of non-standard computation.

if the ruler with which one measures p_j can slip by up to ϵ_i units before p_j is cut.

The combined length $l_1 + l_2$ (which addition the physical system performs accurately[37]), then, lies in the interval $[N - 2\epsilon_i, N + 2\epsilon_i]$ (recall that $N = n_1 + n_2$). If one measures this combined length with additive *output error* $\epsilon_o \geq 0$ (that is, length l is measured as an arbitrary member of $[l - \epsilon_o, l + \epsilon_o]$)—for example to allow for slipping of the ruler as before, though this time during measurement of the combined length—, then we shall obtain a result, R, say, in the interval $[N - \epsilon_\mathrm{b}, N + \epsilon_\mathrm{b}]$, where $\epsilon_\mathrm{b} = 2\epsilon_i + \epsilon_o$.

In order to 'interpret' R (i.e., in order to convert R into a valid answer to the problem being solved), we assume—reasonably—that our physical process is such that values of ϵ_b closer[38] to zero occur with greater probability than those further from zero; we note also that the sought total N is a natural number. Accordingly, we interpret R as the natural number requiring the least ϵ_b, which, in this case, is the nearest natural number $\left\lfloor R + \frac{1}{2} \right\rfloor$ to R (where, if there are two such natural numbers, we arbitrarily choose the greater of the two).

We are guaranteed the correct answer, in that our interpretation of R is N, then, if and only if the interval $[N - \epsilon_\mathrm{b}, N + \epsilon_\mathrm{b}]$ in which we know R to lie is a subset of the interval $\left[N - \frac{1}{2}, N + \frac{1}{2}\right)$ of which elements are interpreted as N. This is the case if and only if both $N - \epsilon_\mathrm{b} \geq N - \frac{1}{2}$ and $N + \epsilon_\mathrm{b} < N + \frac{1}{2}$, which is the case if and only if $\epsilon_\mathrm{b} < \frac{1}{2}$.

Since this constraint on ϵ_b, and hence the constraint on the precision of measurement in the system, does not depend on N (nor, in any other way, on the input values), our intuition should be that the precision complexity of the system is constant. Formally, the region $\left\{ (\epsilon_i, \epsilon_o) \in \mathbb{R}^2 \;\middle|\; \epsilon_i, \epsilon_o \geq 0 \wedge \epsilon_\mathrm{b} < \frac{1}{2} \right\}$ in the plane with axes ϵ_i and ϵ_o (see Fig. 3.2, in which this region is shaded)—that is, the region within which the system is guaranteed to perform correctly the addition of n_1 and n_2—has area $\frac{1}{16}$, regardless of N. The precision complexity, then, which one would intuitively expect to depend solely and inversely on this area—and which in Definition 13 we accordingly define to be one divided by this area—, is 16, again regardless of N.

Note that the system has also an intuitive notion of *space* complexity that is linear in N, since p_1 and p_2, the respective lengths of which total N units, are physically realized (further, there is no additional requirement for storage space: the addends n_j are stored only as lengths p_j, and no intermediate values are used during the calculation). Similarly, drawing a measure along the combined length of p_1 and p_2 takes an amount of time proportional to N; the system has *time* complexity linear in N. (Compare these comments with the machine-independent definitions of time/space complexity, Definition 20.)

The overall complexity—'overall' in a sense that we formalize in Sect. 4.2—of the system, to which its constant precision complexity does not contribute, is therefore *linear* in N.

[37] This reasonable assertion is merely that the combined length of the two pieces is the sum of their individual lengths. Assuming some innocuous and unrestrictive properties of the materials from which the pieces are made (and that their mutual gravitational attraction shortens p_1 and p_2 only negligibly, etc.), this is the case.

[38] In this paragraph, "closer", "further" and "nearest" are defined in terms of the metric giving $|a - b|$ as the distance between real numbers a and b.

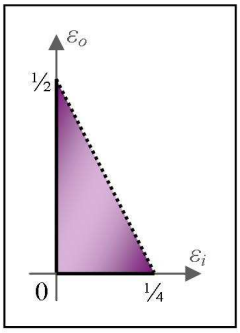

Figure 3.2: The region $\left\{ (\epsilon_i, \epsilon_o) \in \mathbb{R}^2 \;\middle|\; \epsilon_i, \epsilon_o \geq 0 \wedge \epsilon_\mathrm{b} < \frac{1}{2} \right\}$, within which addition is performed by the system of Sect. 3.3.2 *Addition* without error. In this and subsequent such figures, solid boundaries are included in the region and dotted boundaries are not.

Multiplicative Error. Suppose instead that the errors in our physical adding process are better modelled *multiplicatively*, so that (for $j \in \{1, 2\}$) addend n_j is represented by p_j's length l_j, which is in the interval $\left[\frac{n_j}{\epsilon_i}, \epsilon_i n_j\right]$ for some real input error term $\epsilon_i \geq 1$. This occurs, for example, if we measure p_1 and p_2 with a ruler of which the actual length is subject to expansion/contraction by a factor of ϵ_i.

The combined length $l_1 + l_2$, then, is in the interval $\left[\frac{N}{\epsilon_i}, \epsilon_i N\right]$ (recall once more that $N = n_1 + n_2$). If we measure this with multiplicative output error $\epsilon_o \geq 1$ (so that our measurement of a length l lies in $\left[\frac{l}{\epsilon_o}, \epsilon_o l\right]$), then we shall obtain a result R in the interval $\left[\frac{N}{\epsilon_\mathrm{b}}, \epsilon_\mathrm{b} N\right]$, where $\epsilon_\mathrm{b} = \epsilon_i \epsilon_o$.

In order to interpret R, we assume, analogously to the additive case above, that ϵ_b is likely to be small, and we note that N is a natural number. We accordingly interpret R as the natural number requiring the least ϵ_b (making an arbitrary choice—here, the larger—in the event of non-uniqueness): if $R < \sqrt{\lfloor R \rfloor (\lfloor R \rfloor + 1)}$, then we interpret R as $\lfloor R \rfloor$; else, we interpret R as $\lceil R \rceil$.

We are guaranteed the correct answer, in that our interpretation of R is N, then, if and only if the interval $\left[\frac{N}{\epsilon_\mathrm{b}}, \epsilon_\mathrm{b} N\right]$ in which we know that R lies is a subset of the interval $\left[\sqrt{N(N-1)}, \sqrt{N(N+1)}\right)$ of which elements are interpreted as N. This is the case if and only if $\frac{N}{\epsilon_\mathrm{b}} \geq \sqrt{N(N-1)}$ and $\epsilon_\mathrm{b} N < \sqrt{N(N+1)}$, which is the case if and only if $\epsilon_\mathrm{b} < \sqrt{\frac{N+1}{N}}$.

In contrast with the additive case above, this constraint on ϵ_b depends upon N. Specifically, the region $\left\{ (\epsilon_i, \epsilon_o) \in \mathbb{R}^2 \;\middle|\; \epsilon_i, \epsilon_o \geq 1 \wedge \epsilon_\mathrm{b} < \sqrt{\frac{N+1}{N}} \right\}$ (see Fig. 3.3, in which this region is shaded)—within which the system is guaranteed to perform correctly the addition of n_1 and n_2—has area $\int_1^r \frac{r}{\epsilon_i} - 1 \, \mathrm{d}\epsilon_i = r \ln r - (r-1)$ where $r = \sqrt{\frac{N+1}{N}}$. The precision complexity, then, is $(r \ln r - (r-1))^{-1}$, which is bounded below by N^2, as is proven in Lemma 8; that is, the system's precision complexity is at least quadratic.

Further, for the same reason as the former, additive-error system, the latter,

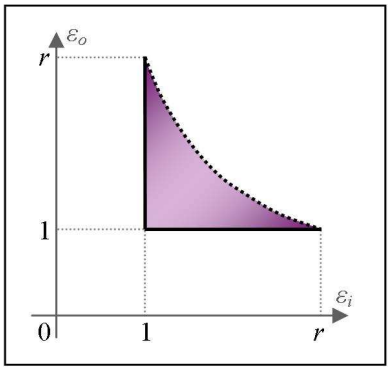

Figure 3.3: The region $\left\{ (\epsilon_i, \epsilon_o) \in \mathbb{R}^2 \;\middle|\; \epsilon_i, \epsilon_o \geq 1 \wedge \epsilon_\mathrm{b} < r \right\}$, within which addition is performed without error by the system of Sect. 3.3.2 *Addition*. Note that r has been exaggerated for visual clarity (whereas, in fact, $1 < r \leq \sqrt{\frac{3}{2}} = 1.224\ldots$ since $2 \leq N < \infty$).

multiplicative-error system has space and time complexity in $\mathcal{O}(N)$. Hence, precision is the most relevant resource (as formalized in Sect. 4.2) when considering this system's overall complexity, which is seen to be at least quadratic in N.

Lemma 8. The area of the region $\left\{ (\epsilon_i, \epsilon_o) \in \mathbb{R}^2 \;\middle|\; \epsilon_i, \epsilon_o \geq 1 \wedge \epsilon_i \epsilon_o < \sqrt{\frac{N+1}{N}} \right\}$ (for $N \in \mathbb{N}$) is bounded above by $\frac{1}{N^2}$. Hence, one divided by this area[39] is bounded below by N^2.

Proof. Let $r = \sqrt{\frac{N+1}{N}}$. The region is a subset of $[1, r]^2$, and so certainly a subset of $\left[1, r^2\right]^2$ (since $N \in \mathbb{N}$, whence $r > 1$, whence $r^2 > r$). Hence, the region's area is bounded above by the area of $\left[1, r^2\right]^2$, which latter area is

$$\begin{aligned} \left(r^2 - 1\right)^2 &= \left(\sqrt{\frac{N+1}{N}}^2 - 1\right)^2 \\ &= \left(\frac{N+1}{N} - 1\right)^2 \\ &= \left(\frac{1}{N}\right)^2 \\ &= \frac{1}{N^2} \;, \end{aligned}$$

as required. □

Aside. Of course, if the lengths of p_1 and p_2 are suitably marked with a *logarithmic* scale, then the combined length (measured on the same scale) offers the input values' *product* rather than their *sum*; this is the principle behind the

[39] This quotient is the precision complexity, with error modelled multiplicatively, of the system of Sect. 3.3.2 *Addition*.

slide-rule's approach to multiplication—see Fig. 3.4. The interested reader is invited to analyze the precision complexity of this implementation of multiplication.

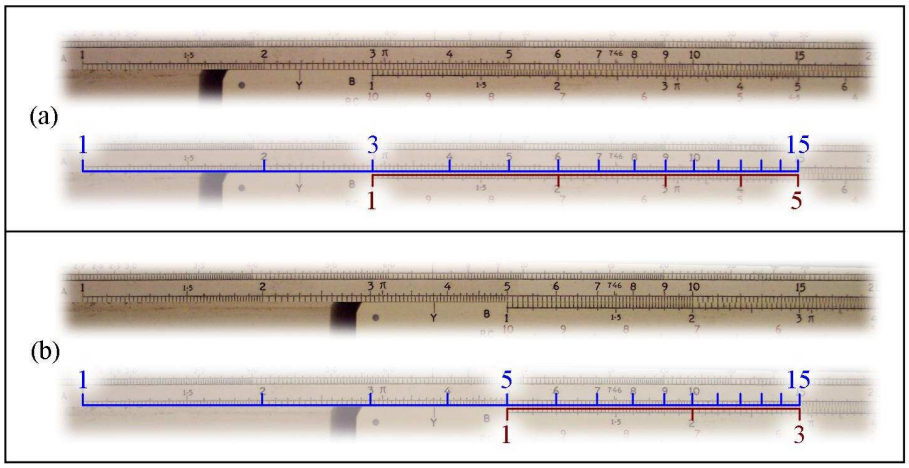

Figure 3.4: Computing with a slide-rule that $3 \times 5 \overset{(a)}{=} 15 \overset{(b)}{=} 5 \times 3$.

Greatest Common Divisor.

Consider now the problem of finding the greatest common divisor (hereafter 'gcd') of two given, positive natural numbers. Recall for comparison that the traditional method, Euclid's algorithm, has time and space complexity polynomial in the size of the input values[40] (and constant precision complexity, since there is no notion of input/output error in the algorithmic model of which it is an instance—see Theorem 1).

The following physical method of solution exploits the behaviour (in particular, the interference) of transverse waves.

We wish to find the gcd of given, positive natural numbers n_1 and n_2. Suppose that our apparatus allows instantiation in the interval $x \in [0, 2\pi)$ of two transverse (sinusoidal, say[41]) waves w_i ($i \in \{1, 2\}$) with respective wavelengths $\lambda_i := \frac{2\pi}{n_i}$; we may model these waves using the graphs $y = \cos\left(\frac{2\pi}{\lambda_i} x\right) = \cos(n_i x)$. Suppose further that these waves are superposed so as to interfere; the resultant superposition—w, say—is given by the graph $y = \cos(n_1 x) + \cos(n_2 x)$.

For $j \in \mathbb{N}$, let M_j be $\left\{ x \in [0, 2\pi) \;\middle|\; \frac{jx}{2\pi} \in \mathbb{Z} \right\} = \left\{ \frac{0\pi}{j}, \frac{2\pi}{j}, \frac{4\pi}{j}, \ldots, \frac{2(j-1)\pi}{j} \right\}$. The cardinality of this set is, by inspection, j.

Now,

- the global maxima of w_i are those values $x \in [0, 2\pi)$ such that $\cos(n_i x) =$

[40] Lemma 11.7 of [103] establishes the former; the latter follows thence and from the observation that space complexity cannot exceed time complexity, simply because moving a Turing machine's read/write head to a new tape cell takes time.

[41] Though, to begin with, we suppose the waves to be sinusoidal (as a nod towards actual implementation with a harmonic oscillator or similar), we below remove this supposition and consider a simplified, triangular waveform.

1, i.e., such that $n_i x$ is a multiple of 2π, i.e., such that $\frac{n_i x}{2\pi}$ is an integer, i.e., such that $x \in M_{n_i}$—*the set of maxima of* w_i *is* M_{n_i};

- therefore, w_i has $|M_{n_i}| = n_i$ maxima in the interval $[0, 2\pi)$; and,
- crucially, the global maxima of w (which have an amplitude twice that of the maxima of w_i) are those points at which both w_1 and w_2 have maxima (that there is indeed such overlap follows from the fact that $M := M_{n_1} \cap M_{n_2} \neq \emptyset$; zero, for example, is in this intersection)—*the set of maxima of* w *is* M.

So as to relate this to the gcd problem, we note the following.

Proposition 16. $M = M_{\gcd(n_1,n_2)}$.

Proof. Suppose that $x \in M_{\gcd(n_1,n_2)}$; then $x \in [0, 2\pi)$, and $\frac{\gcd(n_1,n_2)x}{2\pi} \in \mathbb{Z}$, whence

$$x = \frac{2\pi m}{\gcd(n_1, n_2)} \tag{3.1}$$

for some $m \in \mathbb{Z}$. Since $\gcd(n_1, n_2)$ divides both n_1 and n_2, $\gcd(n_1, n_2) = \frac{n_1}{k_1} = \frac{n_2}{k_2}$ for some $k_1, k_2 \in \mathbb{Z}$. Therefore, $x \overset{(3.1)}{=} \frac{2\pi m k_1}{n_1}$, which is in M_{n_1} since $x \in [0, 2\pi)$ and $mk_1 \in \mathbb{Z}$; similarly, $x = \frac{2\pi m k_2}{n_2} \in M_{n_2}$. Hence, $x \in M$.

Conversely, suppose that $x \in M = M_{n_1} \cap M_{n_2}$; then $x \in [0, 2\pi)$, and $x = \frac{2\pi m_1}{n_1} = \frac{2\pi m_2}{n_2}$ for some $m_1, m_2 \in \mathbb{Z}$. Hence, $\frac{2\pi}{x}$ is a common divisor of n_1 and n_2, and so divides their *greatest* common divisor $\gcd(n_1, n_2)$; say $\gcd(n_1, n_2) = \frac{2\pi m}{x}$ with $m \in \mathbb{Z}$. Then $x = \frac{2\pi m}{\gcd(n_1,n_2)} \in M_{\gcd(n_1,n_2)}$. □

Hence, the set M of maxima (in the interval $[0, 2\pi)$) of w has $|M| = \left|M_{\gcd(n_1,n_2)}\right| = \gcd(n_1, n_2)$ elements. By measuring the *frequency* of the maxima of w—in effect, by counting $|M|$—, then, we find the greatest common divisor of n_1 and n_2.

Aside. This process returns the gcd of n_1 and n_2 because the maxima of wave w_i divide the considered interval into n_i equal sections ($i \in \{1, 2\}$), and so the points of *coincidence* of these waves' respective maxima—i.e., the maxima of w—divide the interval into $\gcd(n_1, n_2)$ equal sections. Compare this situation with the synchronization of planetary orbits or of a clock's hands.

As a concrete example, suppose that we wish to find the gcd of 8 and 6. Then the system instantiates in the interval $x \in [0, 2\pi)$ the waves $y = \cos(8x)$ (shown in Fig. 3.5 in red (■)) and $y = \cos(6x)$ (shown in green (■)), which are superposed (i.e., summed) to give $y = \cos(8x) + \cos(6x)$ (shown in blue (■)).

Then the set $\left\{\frac{0\pi}{8}, \frac{2\pi}{8}, \frac{4\pi}{8}, \ldots, \frac{14\pi}{8}\right\}$ of the maxima of the former constituent wave coincides with that $\left\{\frac{0\pi}{6}, \frac{2\pi}{6}, \frac{4\pi}{6}, \ldots, \frac{10\pi}{6}\right\}$ of the latter at the two[42] points $\frac{0\pi}{8} = \frac{0\pi}{6}$ and $\frac{8\pi}{8} = \frac{6\pi}{6}$ (shown as solid blue (■) circles in Fig. 3.5).

Complexity. That the system as described makes use of *sinusoidal* waves reflects more the way in which the waves may be physically implemented (e.g., with a harmonic oscillator) than an inherently necessary feature of the system's approach to finding gcds. Consequently, we may when considering the system's

[42] That two is the value here is, of course, because $\gcd(8, 6) = 2$.

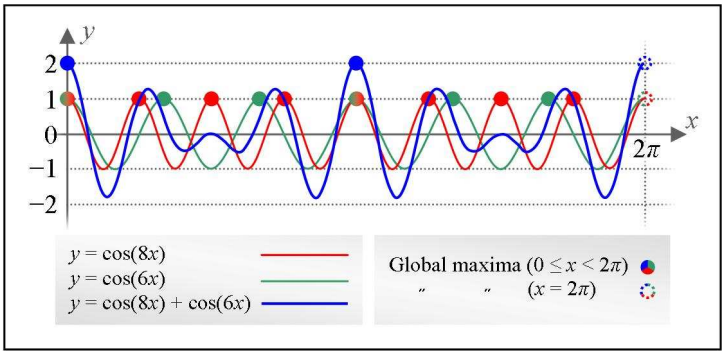

Figure 3.5: Computation, via the superposition of sinusoidal waves, of the greatest common divisor of two given numbers (8 and 6 in this example).

complexity make the simplification of replacing the sinusoidal waves with a more convenient form (and of considering a more convenient interval than $[0, 2\pi)$); we do this as follows.

Suppose that, given input values n_1 and n_2 of which the gcd is sought, the system instantiates in the interval $x \in [0, 1)$ waves w_i given by $y = t_{n_i}(x)$, where, for $n \in \mathbb{N}$, t_n is the triangular-wave function given by

$$t_n(x) = \begin{cases} 2(2nx - \lfloor 2nx \rfloor) - 1 & \text{if } \lfloor 2nx \rfloor \text{ is odd} \\ 2(1 + \lfloor 2nx \rfloor - 2nx) - 1 & \text{if } \lfloor 2nx \rfloor \text{ is even} \end{cases} .$$

See Fig. 3.6, in which t_8 is shown in red (■), t_6 in green (■) and their sum in blue (■).

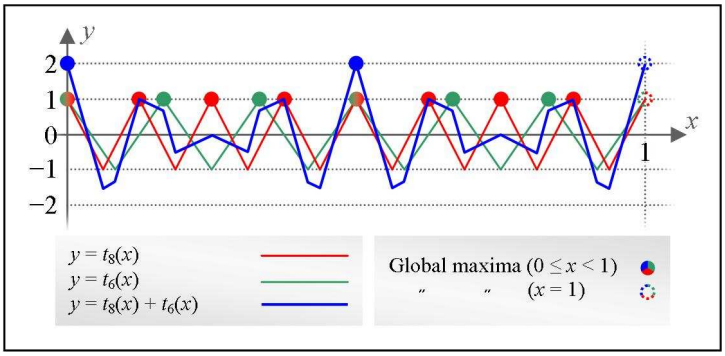

Figure 3.6: Computation, via the superposition of triangular waves, of the greatest common divisor of two given numbers (8 and 6 in this example).

Note that the maxima of these three piecewise-linear waves are identical (but for the x-axis scaling that reflects the change of considered interval) to those of their respective smooth-wave counterparts in Fig. 3.5; in particular, the sum waveform $t_{n_1} + t_{n_2}$ encodes (in its number of maxima) the sought value $\gcd(n_1, n_2)$ (just as w does above).

As a further simplification, we suppose that the input values n_1 and n_2 of

the system satisfy

$$n_1 > n_2 \ . \tag{3.2}$$

It is clear that we lose no generality in supposing that $n_1 \geq n_2$; we may, furthermore, assume that $n_1 \neq n_2$, for else no computational effort (by this system or any other) is necessary: $\gcd(n_1, n_1) = n_1$.

We consider now the *precision complexity* of the system. We claim that, for this system, the constraints (that guarantee that the computation proceeds correctly) on input precision are independent of those on output precision: we may evaluate each individually. Consider first input precision.

Recall that, as input to the system, the positive-integer values n_j ($j \in \{1, 2\}$) are encoded as wavelengths $\frac{1}{n_j}$ of the triangular waves; suppose that adjustment to these values of the actual wavelengths in the system is error prone, and that the physically implemented wavelengths $\frac{1}{l_j}$ suffer, in particular, an *additive* error:[43] the implemented value $\frac{1}{l_j}$ is an arbitrary element of $\left[\frac{1}{n_j} - \epsilon_i, \frac{1}{n_j} + \epsilon_i\right]$ (for some non-negative, real error term ϵ_i), and so $l_j \in \left[\frac{n_j}{1+\epsilon_i n_j}, \frac{n_j}{1-\epsilon_i n_j}\right]$. Suppose further that the system corrects non-integer values of l_j by rounding to the nearest integer l'_j (which, arbitrarily, we take to be the greater of two equally near integers).[44] Now, a small additive change in input values n_j rarely leads to a small additive change in $\gcd(n_1, n_2)$;[45] consequently, it is necessary (and sufficient)—in order that the gcd be correctly computed—that the nearest integer l'_j to l_j coincide with n_j. This occurs precisely when, for each $j \in \{1, 2\}$, the interval $\left[\frac{n_j}{1+\epsilon_i n_j}, \frac{n_j}{1-\epsilon_i n_j}\right]$ in which l_j lies is a subset of the interval $\left[n_j - \frac{1}{2}, n_j + \frac{1}{2}\right)$ of which elements are rounded (during the system's error correction) to n_j, which is the case if and only if both $\frac{n_j}{1+\epsilon_i n_j} \geq n_j - \frac{1}{2}$ and $\frac{n_j}{1-\epsilon_i n_j} < n_j + \frac{1}{2}$, which is the case if and only if $\epsilon_i < \frac{1}{n_1(2n_1+1)}$ (we require also that $\epsilon_i < \frac{1}{n_2(2n_2+1)}$, but this follows automatically from $\epsilon_i < \frac{1}{n_1(2n_1+1)}$ by (3.2)).

The input-precision requirement is that $\epsilon_i < \frac{1}{n_1(2n_1+1)}$.

We consider now output precision, which takes two forms. The ability of the system to find gcds rests on its being able to count the global maxima (in the range $x \in [0, 1)$) of the superposition wave modelled by $y = t_{n_1}(x) + t_{n_2}(x)$ (recall Fig. 3.6 for an example such wave, $t_8 + t_6$); one may reasonably assume, then, that the relevant features of the system (in particular, of the sensor that monitors the superposition wave and detects these maxima) are

- the x-axis resolution[46] ϵ_{o_1}—i.e., the minimum, absolute, x-axis distance between two consecutive global maxima in order that both are registered by the sensor—and

[43] We have in mind an imprecision in setting variable resistors or similar to the correct positions.

[44] This correction may, for example, be achieved by continuously adjusting the wavelength in the range $\left[\frac{1}{l_j+\frac{1}{2}}, \frac{1}{l_j-\frac{1}{2}}\right)$ until the corresponding wave has a maximum at $x = 1$.

[45] This is because of the subtle interaction between the additive and multiplicative structures of the natural numbers, and, specifically, the seeming randomness of the primes.

[46] If x represents time and y the wave's time-dependent amplitude, then the x-axis resolution is the sensor's *temporal* resolution, and the y-axis resolution the sensor's *amplitude* resolution.

- the y-axis resolution[46] ϵ_{o_2}—i.e., the minimum, absolute, y-axis distance between a global maximum and a local but not global maximum in order that the global maximum alone is registered by the sensor.

The x-axis distance between consecutive global maxima of the superposition wave is $\frac{1}{\gcd\left(l_1', l_2'\right)}$, which, assuming the *input*-precision requirement stated above, is equal to $\frac{1}{\gcd(n_1,n_2)}$; required for the correct functioning of the system, then, is that $\epsilon_{o_1} \leq \frac{1}{\gcd(n_1,n_2)}$. Further, the minimal y-axis distance between a global maximum (which has y-value 2) and a local though not global maximum is, by Lemma 9, $2-2\frac{n_1-2\gcd(n_1,n_2)}{n_1} = \frac{2\gcd(n_1,n_2)}{n_1}$; required for the correct functioning of the system, then, is that $\epsilon_{o_2} \leq \frac{2\gcd(n_1,n_2)}{n_1}$. The distances $\frac{1}{\gcd(n_1,n_2)}$ and $\frac{2\gcd(n_1,n_2)}{n_1}$ that must exceed the appropriate sensor resolutions in order that the computation be successful are shown respectively in green (■) and red (■) in Fig. 3.7 (cf. Fig. 3.6).

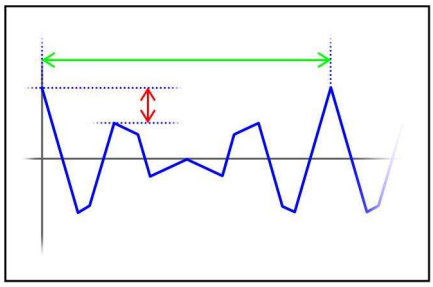

Figure 3.7: The respective distances that must exceed the sensor's x- and y-resolution in order that the gcd computation (in this example, of $\gcd(8,6)$—cf. Fig. 3.6) be successful.

Lemma 9. The non-global, local maxima in $[0,1)$ of superposition wave $t_{n_1}+t_{n_2}$ have values bounded above by $2\frac{n_1-2\gcd(n_1,n_2)}{n_1}$, which bound is attained within this interval.

Proof. Let M_n' be the set $\left\{\frac{0}{n}, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}\right\}$ (for $n \in \mathbb{N}$), and let $t_{m,n}$ be the function that maps x to $t_m(x) + t_n(x)$ (for $m, n \in \mathbb{N}$).

By construction, t_n has maxima over $[0,1)$ precisely at $x \in M_n'$; all are global.

We claim first that, over $[0,1)$, the x-values of *local* maxima of t_{n_1,n_2} are precisely the x-values of the maxima of t_{n_1}, i.e., $x \in M_{n_1}'$. This is for the following reason.

Suppose that x_0 is not a maximum of t_{n_1} (we wish to show that x_0 is not a local maximum of t_{n_1,n_2}), and say that the nearest (in terms of x-axis distance) maximum of t_{n_1} is at x_{m};[47] then the segment of t_{n_1} between x_0 and x_{m} is linear, rather than merely *piecewise* linear (since x_{m} is the *nearest* maximum to x_0), and has gradient $\pm 4n_1$ (by definition of t_{n_1}), with the sign being determined by the fact that $t_{n_1}(x_0) < t_{n_1}(x_{\mathrm{m}})$; hence,

$$t_{n_1}(x_0) = t_{n_1}(x_{\mathrm{m}}) - 4n_1 |x_0 - x_{\mathrm{m}}| \quad . \tag{3.3}$$

[47] Certainly, then, x_0 and x_{m} are distinct.

Now, t_{n_2} is continuous, and the gradient of t_{n_2} is $\pm 4n_2$ almost everywhere[48], whence, for any a and a', $|t_{n_2}(a) - t_{n_2}(a')|$ is bounded above by $4n_2|a - a'|$; in particular, $|t_{n_2}(x_0) - t_{n_2}(x_\mathrm{m})| \leq 4n_2|x_0 - x_\mathrm{m}|$, whence $t_{n_2}(x_0) - t_{n_2}(x_\mathrm{m}) \leq 4n_2|x_0 - x_\mathrm{m}|$ and so

$$t_{n_2}(x_0) \leq t_{n_2}(x_\mathrm{m}) + 4n_2|x_0 - x_\mathrm{m}| \quad . \tag{3.4}$$

Hence,

$$\begin{aligned} t_{n_1,n_2}(x_0) &= t_{n_1}(x_0) + t_{n_2}(x_0) \\ &\overset{(3.3)}{=} t_{n_1}(x_\mathrm{m}) - 4n_1|x_0 - x_\mathrm{m}| + t_{n_2}(x_0) \\ &\overset{(3.4)}{\leq} t_{n_1}(x_\mathrm{m}) - 4n_1|x_0 - x_\mathrm{m}| + t_{n_2}(x_\mathrm{m}) + 4n_2|x_0 - x_\mathrm{m}| \\ &= t_{n_1,n_2}(x_\mathrm{m}) - 4(n_1 - n_2)|x_0 - x_\mathrm{m}| \\ &< t_{n_1,n_2}(x_\mathrm{m}) \quad . \end{aligned}$$

The last inequality holds by (3.2) and because (by Footnote 47) $x_0 \neq x_\mathrm{m}$, whence $4(n_1 - n_2)|x_0 - x_\mathrm{m}|$ is positive.

Further, the uniform, $4n_1$-gradient rise in t_{n_1} as x is altered from x_0 to x_m outweighs any potential fall in t_{n_2}, which has (negative) gradient at most $4n_2 < 4n_1$. Hence t_{n_1,n_2} rises overall as x is altered from x_0 to x_m, and so x_0 is not a local maximum of t_{n_1,n_2}.

Conversely, suppose that x_0 is a maximum of t_{n_1}; we have by a similar gradient argument to that in the previous paragraph that x_0 is a local maximum of t_{n_1,n_2}. Hence,

the set of x-values of local maxima of t_{n_1,n_2} is precisely the set M'_{n_1} of x-values of the maxima of t_{n_1},

as claimed.

We claim next that the local maxima of t_{n_1,n_2} (for which $x \in M'_{n_1}$) *that are not also global maxima* (so $x \notin M'_{n_2}$) satisfy $t_{n_1,n_2}(x) \leq 2\frac{n_1 - 2\gcd(n_1,n_2)}{n_1}$. This is for the following reason.

Suppose that $x \in M'_{n_1} \setminus M'_{n_2}$. Consider a grid of vertical lines with x-values in $M'_k = \left\{\frac{0}{k}, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}\right\}$ where $k = 2\,\mathrm{lcm}(n_1, n_2)$ (we refer below to two subsets $\left\{\frac{0}{k}, \frac{2}{k}, \frac{4}{k}, \ldots, \frac{k-2}{k}\right\}$ and $\left\{\frac{1}{k}, \frac{3}{k}, \frac{5}{k}, \ldots, \frac{k-1}{k}\right\}$ of M'_k, containing respectively 'even-' and 'odd-numerator' values). Now,

$$\text{all maxima of } t_{n_1} \text{ and of } t_{n_2} \text{ are on 'even-numerator' lines,} \tag{3.5}$$

since n_1 and n_2 both divide $\frac{k}{2}$, whence Lemma 10 gives that $M'_{n_1} \cup M'_{n_2} \subseteq M'_{\frac{k}{2}} = \left\{\frac{0}{k}, \frac{2}{k}, \frac{4}{k}, \ldots, \frac{k-2}{k}\right\}$. Further, all minima of t_{n_1} and of t_{n_2} are on (either 'even-' or 'odd-numerator') grid lines, since, by symmetry in x of the definition of t_{n_i}, minima fall midway between maxima. Each linear 'piece' of t_{n_2} climbs from -1 to 1 (or falls from 1 to -1) over $\frac{k}{2n_2}$ grid regions; the values attained by t_{n_2} on grid lines, then, are in $\left\{ j\frac{4n_2}{k} - 1 \;\middle|\; j \in \left\{0, 1, 2, \ldots, \frac{k}{2n_2}\right\}\right\}$. Since $x \notin M'_{n_2}$, $t_{n_2}(x) < 1 = \left(\frac{k}{2n_2}\right)\frac{4n_2}{k} - 1$. Further, $t_{n_2}(x) = \left(\frac{k}{2n_2} - 1\right)\frac{4n_2}{k} - 1$

[48]The measure-zero exception set is M'_{n_2}, in which the gradient is undefined. That t_{n_2} remains continuous at these points is sufficient for our purposes.

only on grid lines one away from a maximum of t_{n_2}; this cannot be the case, for then, by (3.5), x would fall on an 'odd-numerator' grid line, which contradicts that x is a maximum of t_{n_1} (again by (3.5)). So $t_{n_2}(x) \leq \left(\frac{k}{2n_2} - 2\right)\frac{4n_2}{k} - 1$ (and $t_{n_1}(x) = 1$ since $x \in M'_{n_1}$), whence

$$\begin{aligned} t_{n_1,n_2}(x) &= 1 + t_{n_2}(x) \\ &\leq 1 + \left(\frac{k}{2n_2} - 2\right)\frac{4n_2}{k} - 1 \qquad (3.6) \\ &= \left(\frac{k}{2n_2} - 2\right)\frac{4n_2}{k} \\ &= \frac{k - 4n_2}{2n_2}\frac{4n_2}{k} \\ &= 2\frac{k - 4n_2}{k} \\ &= 2\frac{\operatorname{lcm}(n_1,n_2) - 2n_2}{\operatorname{lcm}(n_1,n_2)} \\ &= 2\frac{\gcd(n_1,n_2)\operatorname{lcm}(n_1,n_2) - 2n_2\gcd(n_1,n_2)}{\gcd(n_1,n_2)\operatorname{lcm}(n_1,n_2)} \\ &\overset{(3.7)}{=} 2\frac{n_1n_2 - 2n_2\gcd(n_1,n_2)}{n_1n_2} \\ &= 2\frac{n_1 - 2\gcd(n_1,n_2)}{n_1} \ , \end{aligned}$$

as claimed. (The equality labelled (3.7) follows from the observation—the proof of which we leave to the reader—that, for all $a, b \in \mathbb{N}$,

$$\gcd(a,b)\operatorname{lcm}(a,b) = ab \ .) \qquad (3.7)$$

Finally, this upper bound for t_{n_1,n_2} is attained for all n_1, n_2, as we now show.

Let g denote $\gcd(n_1,n_2)$, and l $\operatorname{lcm}(n_1,n_2)$. By Lemma 1.3.1 of [76], there exist integers i and j such that $in_2 + jn_1 = g$. Therefore, $\frac{in_2-g}{n_1} = -j \in \mathbb{Z}$; i.e.,

$$\frac{in_2 - g}{g}\frac{g}{n_1} \in \mathbb{Z} \ . \qquad (3.8)$$

Let $x' = \frac{in_2}{g}$; then, by (3.8), $x' - 1$ is an integer multiple of $\frac{n_1}{g}$, whence (dividing by l) $\frac{x'}{l}$ is separated by $\frac{1}{l}$ (i.e., by two grid lines) from a multiple of $\frac{n_1}{gl} \overset{(3.7)}{=} \frac{1}{n_2}$. Similarly, since x' is, by definition, an integer multiple of $\frac{n_2}{g}$, $\frac{x'}{l}$ is a multiple of $\frac{n_2}{gl} \overset{(3.7)}{=} \frac{1}{n_1}$.

Adding any integer to $\frac{x'}{l}$ retains this property (namely, that of being a multiple of $\frac{1}{n_1}$ whilst being two grid lines away from a multiple of $\frac{1}{n_2}$); hence, there exists $x \in [0,1)$ with this property. From the above discussion of grid lines (and recalling that $k = 2l$), then, we see that $t_{n_2}(x) = \left(\frac{k}{2n_2} - 2\right)\frac{4n_2}{k} - 1$, whence inequality (3.6) becomes an equality, and t_{n_1,n_2} attains (at x) the value $2\frac{n_1 - 2\gcd(n_1,n_2)}{n_1}$, as required. □

Lemma 10. If a divides b ($a, b \in \mathbb{N}$), then $M'_a \subseteq M'_b$.

Proof. Recall that $M'_n = \left\{\frac{0}{n}, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}\right\}$.

Suppose that a divides b (say $b = al$ with $l \in \mathbb{N}$), and consider $x \in M'_a$. Then $x = \frac{k}{a}$ for some $k \in \{0, 1, 2, \ldots, a-1\}$; i.e., $x = \frac{kl}{b}$. Since $k < a$, $kl < al = b$, and so $x = \frac{kl}{b} \in M'_b$. Hence, $M'_a \subseteq M'_b$. □

These lemmata and the preceding discussion give of the gcd system that

the output-precision requirements are that $\epsilon_{o_1} \leq \frac{1}{\gcd(n_1,n_2)}$ *and that* $\epsilon_{o_2} \leq \frac{2\gcd(n_1,n_2)}{n_1}$.

Hence, we have that the region (in the space spanned by axes ϵ_i, ϵ_{o_1} and ϵ_{o_2}) in which the system correctly evaluates the greatest common divisor of n_1 and n_2 is

$$\left\{ (\epsilon_i, \epsilon_{o_1}, \epsilon_{o_2}) \in \mathbb{R}^3 \;\middle|\; \begin{array}{ll} & 0 \leq \epsilon_i < \frac{1}{n_1(2n_1+1)} \\ \wedge & 0 \leq \epsilon_{o_1} \leq \frac{1}{\gcd(n_1,n_2)} \\ \wedge & 0 \leq \epsilon_{o_2} \leq \frac{2\gcd(n_1,n_2)}{n_1} \end{array} \right\}$$

(see Fig. 3.8). One divided by the volume of this region is

$$\frac{n_1\,(2n_1+1)\gcd(n_1,n_2)\,n_1}{2\gcd(n_1,n_2)} = \frac{1}{2}n_1^2\,(2n_1+1) \in \mathcal{O}\left(n_1^3\right) \quad .$$

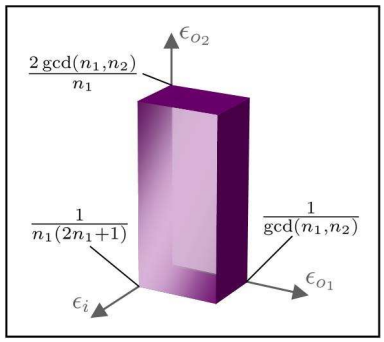

Figure 3.8: The region within which greatest common divisors are found without error.

Aside. We mention in passing the *time* and *space* complexity of the system. The process of wave superposition is (depending upon implementation) virtually instantaneous, and, crucially, takes no longer as n_1 and n_2 acquire more digits[49]; further, larger n_1 and n_2 do not require larger apparatus; the system's wave-superposition stage (wherein the 'processing' occurs), then, has constant time and space complexity. To this we must add the polynomial time/space overheads incurred in converting the addends n_i into the wavelengths of w_i; cf. the harness of Remark 16.

We return to the gcd example in Chap. 4 when motivating the notion of *dominance*.

[49]Supposing that the waves w_i are implemented as electromagnetic radiation, for example, we have that, whilst the wavelength and frequency of w_1 and w_2 depend upon n_1 and n_2, their propagation velocity does not.

3.3.3 Definitions

The definitions in this section are based upon those presented by the author in [18, 28].

Physical Solution Methods.

We introduce in the definitions of Sect. 3.3.3 the notion of *precision*, a resource consumed by certain (in particular, input-/output-error-prone) unconventional computers; we discuss first these computers.

Perhaps surprisingly, our notion of a (physical, error-prone) computation system per se does not, for the purposes of the present project, require formal definition, since there are no set conditions that an object need satisfy in order to constitute such a system. Instead, it is how we view the system[50] as a computing device that is important here. Of interest is the computational system (or process) together with our choice from its physical attributes of what we consider to be the inputs and outputs of the computation. Objects under consideration, then, are physical processes, augmented with two sets—inputs and outputs—of distinguished physical parameters and a rule for interpreting outputs.

Definition 6. A *physical solution method* (or *PSM*) is a tuple $\Phi := (I, \pi, O, \iota)$, where

- I is a tuple (of positive, finite dimension) of pairs (i, V_i), where i is an *input* of Φ and V_i a set of values;
- π is the *process* of Φ;
- O is a tuple (of positive, finite dimension) of pairs (o, V_o), where o is an *output* of Φ and V_o a set of values; and
- ι is the *interpretation function* of Φ, which maps each output value of Φ (see Definition 7) to an output value.[51]

Remark 18. The intuition behind Definition 6 is as follows.

A physical computation system has parameters that admit alteration so as to supply input to the system—the parameters' values encode the desired input value. For coordinate (i, V_i) of I, i acts as an input parameter (the angle of a mechanical shaft or the concentration of a chemical solution, for instance), and V_i as the set ($[0, 2\pi)$ or $[0, 1]$, for instance) of values to which i can be set.

After setting the input parameters, a process—typically presented as a description of the physical apparatus[52] along with an algorithm/program consisting of physical instructions—is carried out;[53] π models this process.

[50] We pay particular regard to the system's *input* and *output* processes, since this is where the error, and, hence, non-trivial precision complexity, arise.

[51] In practice, ι is used to map *measured* output values to *interpreted* output values—see Definition 10.

[52] We do not assume that the same apparatus can universally accommodate all input values; larger inputs may require larger apparatus with more physical storage, for example.

[53] The system may execute the process automatically, or user intervention, uniquely defined by the description of the process, may be required. Further, what we present here as a computation using a physical solution method may provide only an *intermediate* result, which is used as (possibly partial) input to subsequent computations; in this way, we can accommodate interactive computation.

Output is then taken from the system in the form of a measurement of certain parameters; each such is modelled by a coordinate (o, V_o) of O: o models the mensurable parameter, and V_o the set of values that o may take.

Finally, the output value tuple measured by the user is converted by ι into an 'interpreted' output value tuple. This conversion can be thought of as a form of error correction: the interpretation function takes *arbitrary* output and interprets it as *corrected* output, typically based upon a priori knowledge about the form of the expected output.[54]

Note that "physical instruction", "parameter" and other terms used in this remark are neither formally defined nor used in Definition 6. This renders the class of PSMs sufficiently wide to include instances of many models of computation. We wish, in particular, to consider abstract properties of input and output (notably their susceptibility to error) without restricting the way in which 'processing' by the system—whatever form that may take—converts the former into the latter.

Definition 7. Given a PSM $\Phi = (I, \pi, O, \iota)$ with $I = \big((i_1, V_{i_1}), \ldots, (i_p, V_{i_p})\big)$, an *input value* of Φ is an assignment of 'allowed' values to the inputs of Φ; that is, a tuple $\big(v_{i_1}, \ldots, v_{i_p}\big) \in V_{i_1} \times \ldots \times V_{i_p}$ of values, where to i_j is considered to be assigned the value v_{i_j}.

Similarly, supposing that $O = \big((o_1, V_{o_1}), \ldots, (o_q, V_{o_q})\big)$, an *output value* of Φ is an assignment of 'allowed values' to the outputs of Φ; that is, a tuple $\big(v_{o_1}, \ldots, v_{o_q}\big) \in V_{o_1} \times \ldots \times V_{o_q}$ of values, where to o_j is considered to be assigned v_{o_j}.

Definition 8. For a PSM $\Phi = (I, \pi, O, \iota)$, define the *computation relation* $\smile_\Phi$ to be the set of pairs (x, y) such that, on assigning to Φ the input value x and executing process π of Φ, (uninterpreted) output value y can result. As is normal with relations, write '$x \smile_\Phi y$' for '$(x, y) \in \smile_\Phi$'.

Definition 9. A PSM Φ is said to be *deterministic* (a *D*PSM) if, for every input value x of Φ, there exists a unique output value y such that $x \smile_\Phi y$; then $\smile_\Phi$ may be treated as a function. '*Non-deterministic* PSM' (or '*N*PSM') is another term for PSM, whether deterministic or not; we use 'NPSM' rather than 'PSM' when we wish to emphasize that a machine is not necessarily deterministic.

Aside. Note that an arguably more natural definition would have labelled as DPSMs the deterministic PSMs, and as NPSMs the *strictly* non-deterministic PSMs (rather than the *not-necessarily*-deterministic PSMs, as in Definition 9). However, we favour Definition 9 because of its respecting the analogy with the standard definition (see for example [37]) of *non-deterministic Turing machines*—which should properly be called not-necessarily-deterministic Turing machines!—, which notion includes as a subset *deterministic* Turing machines. Further, there is a sense in which the choice makes no difference: by Lemma 11 and Remark 19, for every DPSM, there exists a *strictly* non-deterministic PSM that performs the same computation and, further, consumes the same resources; therefore, the class of problems solved/computations performed by NPSMs is independent of whether 'NPSM' denotes 'strictly non-deterministic PSM' or 'not-necessarily-deterministic PSM'.

[54] For example, if the computation is such that the expected answer is an integer, then the measured output value, a real number, say, may by way of interpretation be rounded off to the nearest integer.

Lemma 11. For each DPSM, there exists a (strictly) non-deterministic PSM that performs the same computation.

Proof. We construct a non-deterministic PSM Ψ that acts as the given, deterministic PSM Φ, but for its exhibiting non-determinism (which is ignored) in an additional output parameter.

Consider DPSM $\Phi = \left(I, \pi, \left(\left(o_1, V_{o_1}\right), \ldots, \left(o_q, V_{o_q}\right)\right), \iota\right)$. Let Ψ be the PSM $\left(I, \pi', \left(\left(o_1, V_{o_1}\right), \ldots, \left(o_q, V_{o_q}\right), \left(o, \{0,1\}\right)\right), \iota'\right)$, where

- π' performs the process π, followed by a non-deterministic assignation of either 0 or 1 to the additional output parameter o; and
- ι' maps $\left(v_{o_1}, \ldots, v_{o_q}, b\right) \in V_{o_1} \times \ldots \times V_{o_q} \times \{0,1\}$ to $(w_1, \ldots, w_q, 0)$ (regardless of the value of b), where $(w_1, \ldots, w_q) = \iota\left(v_{o_1}, \ldots, v_{o_q}\right)$.

Fix x, an input value for Ψ, and suppose that (uninterpreted) output value y is such that $x \smile_\Psi y$. By construction, then, y without its final coordinate (call this y') satisfies $x \smile_\Phi y'$, and is hence unique given x, since Φ is deterministic. Further, because of the non-deterministic assignation of o, y' with *either* 0 or 1 appended as a final coordinate can serve as an output value from Ψ given x: for each input value x, there exist exactly two output values y such that $x \smile_\Psi y$; Ψ is, therefore, a strictly non-deterministic PSM. Non-determinism notwithstanding, all information present in any y' such that $x \smile_\Phi y'$ is also present in any y such that $x \smile_\Psi y$ (we merely ignore the final parameter of y in order to retrieve y'). So, we have exhibited an NPSM with the same behaviour as our given DPSM, as required. □

Remark 19. Note also that Φ and Ψ in the above proof consume asymptotically equal resources. The only differences in resource consumption are that Ψ must spend a constant amount of run-time to assign non-deterministically a bit, and must use a constant amount of storage space to store this bit (the system may also consume, for example, a constant amount of energy during this time). These constant additional overheads leave the complexity functions' $\mathcal{O}$-behaviour unchanged.

Error; Precision Complexity.

Recall from Remark 18 the process whereby a PSM performs a computation. We now elaborate on this process, and model the fact that, in practice, errors can be introduced into the system during both the application of input values and the measurement of output values. We use *non-determinism* to model these errors: for example, a value x, as perturbed by some additive error with parameter $\epsilon \geq 0$, may be rendered as a non-deterministic, arbitrary element of $[x - \epsilon, x + \epsilon]$ (if the error were instead multiplicative, with parameter $\epsilon \geq 1$, say, then the perturbed value would be a non-deterministic choice from $\left[\frac{x}{\epsilon}, \epsilon x\right]$—recall and contrast Sects. 3.3.2 *Addition—Additive Error* and 3.3.2 *Addition—Multiplicative Error*).

Definition 10. Computation via PSM Φ consists of four stages (say that $\Phi = (I, \pi, O, \iota)$ with $I = \left(\left(i_1, V_{i_1}\right), \ldots, \left(i_p, V_{i_p}\right)\right)$ and $O = \left(\left(o_1, V_{o_1}\right), \ldots, \left(o_q, V_{o_q}\right)\right)$):

- INPUT. The user attempts to apply to Φ his *intended input value*—the input value $x := \left(v_{i_1}, \ldots, v_{i_p}\right)$ at which he wishes to evaluate function $\smile_\Phi$ (or, if Φ is non-deterministic, for which he wishes to find some y such that $x \smile_\Phi y$)—by adjusting the inputs of Φ. Due to lack of precision in this adjustment, however, the intended input value x may not coincide with the *implemented input value* $x' := \left(v'_{i_1}, \ldots, v'_{i_p}\right)$—the actual input value received by the PSM. There is (assuming that the PSM is of practical use), however, a relationship between each v_{i_j} and v'_{i_j} in terms of some real error term ϵ_{i_j}—the *specific input error* for i_j.[55] Define the *generic input error* (or, simply, *input error*) of Φ to be the tuple $\epsilon_I := \left(\epsilon_{i_1}, \ldots, \epsilon_{i_p}\right) \in \mathbb{R}^p$. Given an intended input value, then, the corresponding implemented input value is not necessarily uniquely determined, but is at least bounded in some sense by the input error (recall the above comments concerning the non-deterministic modelling of noise). The error introduced whilst inputting a value may, therefore, be modelled by the *input error relation* R_{ϵ_I}, a relation, parameterized by ϵ_I, between intended and implemented input values.

- EXECUTION. The process π of Φ is then executed.

- MEASUREMENT. Next, the user attempts to measure the output value from Φ. Again due to lack of precision, this time during measurement, the *true output value*[56] $y' := \left(v'_{o_1}, \ldots, v'_{o_q}\right)$—that supplied by Φ—and the *measured output value* $y'' := \left(v''_{o_1}, \ldots, v''_{o_q}\right)$—that measured by the user—may not coincide. Again, however, there is a relationship between each v'_{o_j} and v''_{o_j} in terms of some real error term ϵ_{o_j}—the *specific output error* for o_j. Let the *generic output error* (or, simply, *output error*) of Φ be the tuple $\epsilon_O := \left(\epsilon_{o_1}, \ldots, \epsilon_{o_q}\right) \in \mathbb{R}^q$, and let the *error*[57] of Φ be the pair $\epsilon := (\epsilon_I, \epsilon_O) \in \mathbb{R}^p \times \mathbb{R}^q$.[58] Given a true output value, then, the corresponding measured output value is not necessarily uniquely determined, but is at least bounded in some sense by the output error (recall once more the modelling of noise as non-determinism). The error introduced during measurement of an output value may, much as during input, be modelled by the *output error relation* R_{ϵ_O}, a relation, parameterized by ϵ_O, between true and measured output values.

- INTERPRETATION. Finally, the user applies the interpretation function of Φ to the measured output value y'' to find the *interpreted output value*

[55] For example, it may be the case that v_{i_j} and v'_{i_j} differ by no more than specific input error $\epsilon_{i_j} \geq 0$; cf. n_j and l_j in Sect. 3.3.2 *Addition—Additive Error*.

[56] Note that this value y' satisfies $x' \smile_\Phi y'$. Tautologically, the physical system computes *with perfect accuracy* whatever relation it computes, even if the user's ability to input and measure values is imperfect.

[57] For our purposes, the error of a PSM is an independent variable—we are free to alter the error, and wish to consider those values of the error for which the system performs the correct computation (at least given input of a certain size). In this footnote, we detect the germ of a *complexity measure*...

[58] We sometimes view error ϵ as an element of the space $\mathbb{R}^{p+q}$, using the obvious function $((a_1, \ldots, a_p), (b_1, \ldots, b_q)) \mapsto (a_1, \ldots, a_p, b_1, \ldots, b_q)$ from $\mathbb{R}^p \times \mathbb{R}^q$ to $\mathbb{R}^{p+q}$. In such contexts, then, ϵ is the tuple concatenation, rather than the pair, of ϵ_I and ϵ_O.

$y := \iota(y'')$.[59]

The 'flow of computation' in Definition 10, then, is as follows: intended input value $\overset{\text{imprecision}}{\mapsto}$ implemented input value $\overset{\pi}{\mapsto}$ true output value $\overset{\text{imprecision}}{\mapsto}$ measured output value $\overset{\iota}{\mapsto}$ interpreted output value.

Definition 11. Say that intended input value x (Φ, ϵ)-*yields* interpreted output value y if and only if there exist x', y' and y'' such that $x \, R_{\epsilon_I} \, x'$, $x' \smile_\Phi y'$, $y' \, R_{\epsilon_O} \, y''$ and $\iota(y'') = y$ (where $\epsilon = (\epsilon_I, \epsilon_O)$).[60] Define the *yield relation* $\frown_{\Phi,\epsilon}$ to be $\{ (x, y) \mid x \ (\Phi, \epsilon)\text{-yields } y \}$, and write '$x \frown_{\Phi,\epsilon} y$' for '$(x, y) \in \frown_{\Phi,\epsilon}$'.

Remark 20. Note that the 'smile' notation $\smile$ denotes the happy situation in which computation proceeds in a perfect, error-free way: it relates unperturbed, abstract input/output values as may be specified in a mathematical problem's description. The 'frown' notation $\frown$, on the other hand, denotes the unhappy (but more interesting!) situation in which input/output error may be present: it relates possibly noisy input and output values that may not give the correct answer to a mathematical problem.

Intuitively, if the specific input and output errors of a PSM Φ constrain sufficiently tightly that intended and implemented input, and also true and measured output, values be close together,[61] then the error in the system is sufficiently small that some intended input and corresponding yielded interpreted output values will be related by $\smile_\Phi$—these small errors introduced during input and measurement are successfully corrected during interpretation. We are, in particular, interested in the set of errors ϵ that guarantee that a given input value will be related by $\smile_\Phi$ to any corresponding (Φ, ϵ)-yielded interpreted output value; the behaviour (usually, the dwindling) of this set of corrigible errors as the input size increases is the basis of our notion of precision complexity. These ideas are now formalized.

Definition 12. A *size function* for PSM Φ is a map from the set of input values of Φ to the set $\mathbb{R}^+$ of non-negative real numbers.[62]

Definition 13. Let Φ be PSM (I, π, O, ι) with $I = \left((i_1, V_{i_1}), \ldots, \left(i_p, V_{i_p}\right) \right)$ and $O = \left((o_1, V_{o_1}), \ldots, \left(o_q, V_{o_q}\right) \right)$, let σ be a size function for Φ, and let x be an input value for Φ.

- An error $\epsilon \in \mathbb{R}^{p+q}$ is said to be *precise for* x if, for all y such that $x \frown_{\Phi,\epsilon} y$, it is the case that $x \smile_\Phi y$.[63]
- Let $\mathcal{E}_\Phi(x)$ denote the set $\{ \epsilon \in \mathbb{R}^{p+q} \mid \epsilon \text{ is precise for } x \}$ of errors precise for x.

[59] This interpretation computation may well be achieved in practice via Turing machine—cf. Remark 16.

[60] Intuitively, an attempt by the user to input x, execute the process, and measure and interpret the output can result in the output y if and only if x (Φ, ϵ)-yields y.

[61] This can happen by the specific errors being close to zero if the errors modelled are additive, one if multiplicative, and so on. The existence of an identity element (0, 1, etc.) for the operation in question (+, ×, etc.) allows for singleton, measure-zero intervals ($[n - \epsilon, n + \epsilon]$, $\left[\frac{n}{\epsilon}, \epsilon n\right]$, etc.); other 'non-operation' types of error may have similarly trivial non-determinism during input and measurement.

[62] For a discussion of *general* size functions, as opposed to those specifically for PSMs, see Sect. 3.4.3 *Size Functions*.

[63] Intuitively, then, an error is precise for x if and only if it is sufficiently 'small' that, at least on input x, it is corrected during interpretation.

- Let $\mathcal{V}_\Phi(x)$ denote the Euclidean, $(p+q)$-dimensional volume[64] of $\mathcal{E}_\Phi(x)$. Thus, $\mathcal{V}_\Phi(x) \in [0, \infty]$.
- Let the *precision required by* Φ *given* x be $P_\Phi(x) := \left\lfloor \frac{1}{\mathcal{V}_\Phi(x)} \right\rfloor \in \mathbb{N} \cup \{\infty\}$.[65]
- Define Φ's *precision complexity relative to* σ to be the function $P^*_{\Phi,\sigma} \colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ given by

$$P^*_{\Phi,\sigma}(n) = \sup \{ P_\Phi(x) \mid \sigma(x) = n \} \quad .$$

When the choice of size function and/or PSM is understood, references thereto (e.g., subscripts '$_\sigma$' and '$_\Phi$') may be omitted.

Remark 21. P_Φ is our illustrative example of a (non-standard) *resource*. See the formalization of this concept below (Sects. 3.4 and 4.5.4).

Remark 22. There is nothing particular to the resource P of *precision* that prompts us to define the corresponding measure P^* of complexity as in Definition 13; we may generalize this construction so as automatically to obtain a complexity function given an arbitrary computational resource. See Sect. 3.4.3.

Remark 23. In many practical cases, we have the following. If input values x_1 and x_2 are such that $\sigma(x_1) \leq \sigma(x_2)$, then, for a 'practically useful' PSM $\Phi = (I, \pi, O, \iota)$ and 'sensible' size function, we have that $\mathbb{R}^{\dim I + \dim O} \supseteq \mathcal{E}_\Phi(x_1) \supseteq \mathcal{E}_\Phi(x_2) \supseteq \emptyset$, whence (taking volumes) $\infty \geq \mathcal{V}_\Phi(x_1) \geq \mathcal{V}_\Phi(x_2) \geq 0$ and (taking floored reciprocals) $0 \leq P_\Phi(x_1) \leq P_\Phi(x_2) \leq \infty$. P^*_Φ is, in these 'common' cases, a non-decreasing function of the size of an input value.

(We make no attempt to formalize "practically useful", "sensible" or "common", but instead include this remark so as to give an intuitive feel of the 'direction' in which precision increases. Cf. the typical situation (though by no means necessity) of a Turing machine's requiring more time as larger input values are supplied.)

Remark 24. Note that our definition of the precision required by Φ given x—namely, the floor of one divided by the *volume* of $\mathcal{E}_\Phi(x)$—is by no means the uniquely most obvious choice. Another promising candidate is the floor of one divided by the *most quickly shrinking side*[66] of the minimal hypercuboid that bounds $\mathcal{E}_\Phi(x)$; indeed, this latter definition seems more consistent with Definition 23, and seems better to reflect precision's not superficially being inherently multiplicative in the dimensions of $\mathcal{E}_\Phi(x)$. However, our adoption in Definition 13 of the former choice reflects the intention that precision capture 'lack of robustness' of a computation, and hence be reciprocal to the 'size' (i.e., volume rather than maximal length) of the set $\mathcal{E}_\Phi(x)$ of corrigible errors. An extreme (though unlikely) motivating case for this view sees $\mathcal{E}_\Phi(x)$ take the form of a space-filling curve.

[64] For our purposes, the choice of measure in terms of which multidimensional volume is defined is unimportant; we arbitrarily choose Lebesgue measure. Regardless of dimension, we occasionally use 'volume' to denote whichever is appropriate of length, area, volume, hypervolume, etc.

[65] For this purpose, we use the standard extensions $\frac{1}{0} = \infty$ and $\frac{1}{\infty} = 0$ of real-number division.

[66] That is, the side one divided by which (viewed as a function of the size of x) $\mathcal{O}$-exceeds one divided by each other side (viewed similarly), assuming that such exists or else adopting some reasonable tie-breaking convention.

It can be seen that the conclusions of Sect. 3.3.2—most notably that the additive- and multiplicative-error systems of addition have precision-complexity functions respectively constant and at-least-quadratic in $n_1 + n_2$, and that the gcd system has precision complexity cubic in n_1—remain valid under the formalization (Definition 13) of precision complexity[67] (apart from the fact that, in the examples of Sect. 3.3.2, we avoid the obfuscating detail of taking floors; but this, of course, does not affect the asymptotic behaviour of the systems' precision complexity).

Aside. In fact, there is a second way in which the precision-complexity analysis of the systems of Sect. 3.3.2 differs from the prescription of Definition 13. Specifically, where each of two input parameters is prone to the *same* error (as is the case with the addition and greatest common divisor systems, of each of which each input value is corrupted in the same way and by the same error term as the other), we model the corresponding error term as a single axis ϵ_i—rather than a separate axis ϵ_{i_j} for each corrupt parameter, as is suggested by Definitions 10 and 13—in the space of errors.

The modification necessary for the definitions to be unambiguous in this respect, then, is the inclusion of a criterion that makes precise in which situations one of a pair of error terms can and should be neglected. We suggest that such a criterion can be formulated using the notion of the error terms' *joint entropy*, though defer further detail to future work.

Theorem 1. An instance (e.g., a Turing machine) of a standard model of computation can be expressed as a PSM Φ, in that the relation between inputs and outputs of the instance is $\frown_{\Phi,\epsilon}$ (where the choice of ϵ is unimportant). Further, $P^*_{\Phi,\sigma}(n) = 0$ for all n and for all σ.

Proof. Note that, in a standard model of computation, there is assumed to be no error during input and measurement (and hence no need for interpretation): the intended input value is faithfully passed to the algorithm, which supplies an output that can be accurately read.[68]

This situation can be expressed in the more general, error-accommodating framework of PSMs defined above by letting the input/output error relations and the interpretation function be identity maps (and by letting the process of the PSM, $\Phi = (I, \pi, O, \iota)$, say, be a physical implementation—a digital computer with sufficient physical storage and running an appropriate program, for example—of the standard-computer instance itself).

Since, in particular, the input/output error relations R_{ϵ_I} and R_{ϵ_O} are chosen to be identity maps for *any* generic errors ϵ_I and ϵ_O, *every* error is precise for all input values. Hence, for all x, $\mathcal{E}_\Phi(x) = \mathbb{R}^{\dim I + \dim O}$, which has infinite volume, and so the precision complexity of a Turing machine (or similarly standard computer) is constantly $\sup\left\{\left\lfloor \frac{1}{\infty} \right\rfloor\right\} = 0$. □

[67] Note that these specific precision-complexity functions are essentially viewed as functions of input values rather than *sizes* of input values; we have, therefore, implicitly used an identity size function.

[68] We recall the same sentiment expressed in [128].

> "What 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."

Where convenient in what follows, we assume that instances of standard models of computation are presented as Turing machines; this is in light of the original and extended Church-Turing theses and, as a specific example of the latter thesis, Remark 25.

Definition 14. In light of Theorem 1, we define a *standard solution method* (or *SSM*) to be a PSM with identity input/output error relations and interpretation function.

Time Complexity; Space Complexity.

To exploit fully the fact (proven in Theorem 1) that Turing machines and similar (viewed as SSMs) are a special case of PSMs, we define for PSMs measures of *time* and *space* complexity; naturally, when the computer in question is an SSM, we wish these measures to agree with the existing, standard-complexity-theory definitions. This extension[69] of the class of computers of which we can ascertain such complexity measures aids fair comparison of SSMs with other PSMs, as well as meaningful comparison of different measures (time, space, precision, etc.) of complexity.[70]

Definition 15. Let Φ be a PSM, σ a size function, and x an input value.

- Define the *run-time* $T_\Phi(x) \in \mathbb{N} \cup \{\infty\}$ *of* Φ *given* x to be the integer part of the (possibly infinite) time taken from (a) supply to Φ of x until (b) receipt from Φ of an (interpreted) output value.[71]
- Define Φ's *time complexity relative to* σ to be the function $T^*_{\Phi,\sigma}\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ with

$$T^*_{\Phi,\sigma}(n) = \sup\{\, T_\Phi(x) \mid \sigma(x) = n \,\} \quad .^{72}$$

Remark 25. Recall, for example from Definition 2.5 of [103], the traditional notion of time complexity: run-time of a Turing machine given some input is the (possibly infinite) number of time steps in the computation of the machine on the input until a halting state is reached, and the machine's time complexity evaluated at n is, as here, the supremum over inputs of size n of run-time (see also Sect. 3.4.3).

Other standard models—the random-access machines of Sect. 2.6 of [103], for example—have similar notions of time complexity; further, for each random-access machine, there is an equivalent[73] Turing machine with time complexity only polynomially different from that of the random-access machine, and vice versa.

[69] Since they are applicable to virtually all computational models, we extend these notions to yet further generality in Sect. 3.5.1 *Time and Space*.

[70] Such comparisons are the focus of Chap. 4.

[71] The units in which this time is measured are of no importance, since we primarily consider the asymptotic behaviour of resource T and its complexity function T^*. For the sake of run-time's being well-defined, however, let us take *seconds* as our unit.

[72] As with Definition 13, 'relative to σ' and the subscripts '$_\Phi$' and '$_\sigma$' are sometimes omitted when understood.

[73] Two computers, algorithms or similar are *equivalent* if they compute the same function/relation.

Theorem 2. For all functions f, $\mathcal{O}(f)$ contains the time complexity (according to Definition 15) of an SSM if and only if it contains the traditionally defined time complexity (according to Definition 2.5 of [103], say) of the corresponding[74] Turing machine.

Proof. Fix a function f.

Suppose that SSM Φ is such that $T^*_\Phi \in \mathcal{O}(f)$. Since the process of Φ is a direct implementation of the corresponding Turing machine M, every operation—of which there are only finitely many—achievable by Φ in one second can be performed by M in a finite number of steps. Let s be the (constant) maximum of these finitely many finite numbers of steps. Then the run-time of M given some input is no more than s times the run-time (in seconds) of Φ given the same input; hence, M has time complexity in $\mathcal{O}(sf) = \mathcal{O}(f)$.

Similarly but conversely, suppose that a Turing machine M has time complexity in $\mathcal{O}(f)$. Let Φ be the corresponding SSM, which is an implementation (on a digital computer, for example) of M; each 'atomic operation'—of which there are only finitely many—that M can perform in one time step is implementable by the process of Φ in finite time. Let t be the (necessarily finite, constant) maximum of these finitely many finite times. Then the run-time of Φ given some input is no more than t times the run-time of M given the same input; hence, $T^*_\Phi \in \mathcal{O}(tf) = \mathcal{O}(f)$. □

Definition 16. Let Φ be a PSM, σ a size function, and x an input value.

- Define the *space* $S_\Phi(x)$ *needed by* Φ *given* x to be the physical, three-dimensional volume required by Φ to perform its computation with input x.[75]
- Define Φ's *space complexity relative to* σ to be the function $S^*_{\Phi,\sigma}\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ with

$$S^*_{\Phi,\sigma}(n) = \sup\{\, S_\Phi(x) \mid \sigma(x) = n \,\} \quad .^{76}$$

Remark 26. Compare with Definition 16 the traditional notion of space complexity (given, for example, in Definition 2.6 of [103]): the space required by a Turing machine on an input is the number of (distinct) tape cells used by the machine in processing the input. Space complexity evaluated at n is, as here, the supremum over inputs of size n of required space.

Theorem 3. For all functions f, $\mathcal{O}(f)$ contains the space complexity (according to Definition 16) of an SSM if and only if it contains the traditionally defined space complexity (according to Definition 2.6 of [103], say) of the corresponding Turing machine.

Proof. Note that there exists a multiplicative constant that bounds the size of each of a Turing machine cell and a unit of physical data storage in terms of the other[77]; then the proof is as that of Theorem 2. □

[74] The correspondence here is in the sense of Theorem 1

[75] Similarly to Definition 15, the units in which this volume is measured are not important, since we primarily consider asymptotic behaviour. So that space is well-defined, however, let us measure in *cubic metres*.

[76] 'Relative to σ' and the subscripts '$_\Phi$' and '$_\sigma$' are sometimes omitted when understood.

[77] This follows from the fact that there is not only a practical, but also theoretical, limit on the density of data storage (that is, the number of bits that can be stored per cubic meter, say)—see [13].

We see, then, that the (traditionally defined) time- and space-complexity functions of a Turing machine or similar are, up to $\mathcal{O}$-notation, the same as the time- and space-complexity functions (according to Definitions 15 and 16) of the machine's PSM formulation: the class of SSMs is embedded in the class of PSMs, and the time- and space-complexity functions of the latter extend those of the former.

3.3.4 Precision in the Literature

We list now some considerations of precision, in the context of computational complexity, made in the literature (some are discussed in greater detail elsewhere in the present work). The differences (of which some are mentioned in the following list) between these items and the present work hint at the novelty of the framework of computation and complexity that we introduce.

- An exploration ([8,9,12]) of the additional computing power[78] gained from augmenting the Turing machine with position-measuring oracles (based upon particle-scattering to find a wedge's vertex: each face meeting at the vertex reflects the particles in a different direction, and to a different sensor, betraying information about the vertex's position). It is assumed that the particles do not alter the position of the wedge; indeed they do not *a great amount*, but one wishes to distinguish arbitrarily close positions, which entails their not altering the wedge position *at all*. Given either error-free or arbitrarily precise (that is, correct to within any given $\epsilon > 0$) particle-cannon positioning, machines so augmented compute all of $\mathsf{P/poly}$, which contains the Halting set; with fixed-error positioning (correct to within some fixed $\epsilon > 0$), they compute $\mathsf{BPP//log}*$, which includes non-recursive sets; both classes, then, indicate super-Turing power. Consideration of precision in [9, 12] has as its focus the particle cannon's position; wedge-vertex position (in the exactitude of which lies the increase in computational power) warrants similar analysis, and is accordingly addressed in [8]. (We recall also an extension in [11] of these ideas from the mechanical to the electrical.)

- Presentation ([30]) of a Turing-machine-like framework to accommodate real-number computations, rather than traditional, discrete (say, without loss of generality, $\{0, 1\}$-based) computations, though without focus on precision.[79]

- Discussion ([110]) of the undecidability of the ray-tracing problem, which benefits from formalization (particularly of precision issues) in the context of this project—see Sect. 5.2.

- Contrasting views ([44, 96]) of the significance for the Kochen-Specker theorem of precision's being necessarily finite in practice, which, again, may benefit from consideration in the present context—see Sect. 5.4.

[78] This is expressed in terms of 'advice classes' $\mathcal{B}/\mathcal{F}$, which allow non-uniformity in the sense that computational processes are allowed to be dependent on input size (or, equivalently, can make use of 'advice strings' depending only upon the input size).

[79] In fact, [30] largely overlooks not only the precision costs incurred *during computations* involving real numbers, but also the space costs associated with real numbers' *storage*, deeming these latter costs to be constant. Storage of real numbers is more thoroughly treated in [101], wherein ideas based upon Dedekind cuts are used.

- Presentation ([87, 88]) of a quantum process that purports to decide a Turing-undecidable problem; this is yet another test case discussed via formalization (especially of precision issues) according to the notions of this project—see Sect. 5.3.

- An investigation ([133,134]) into an optical model of computation (and the complexity of instances thereof), though avoiding some issues of precision by implicitly using a trade-off with space (in this case, number of pixels)—see Sect. 3.6.1 *Optical Computers*.

- A comprehensive study ([39]) of continuous-variable quantum systems—see Sect. 3.6.2 *Continuous-Variable Quantum Computers*.

- A study ([128]) of analogue complexity, in which precision of measurement is seen as a factor constraining the set of problems that can be solved, rather than—as here—a dependent-variable property of problem instances and hence a resource type.

- An exploration ([65]) of computers' ability to simulate physics—including quantum phenomena—, with some consideration of precision issues such as the discrete representation of space-time.

- A strengthening ([71]) of the foundations of a specific model of analogue computer, in particular to accommodate real-number manipulations.

- Acknowledgement ([102]) that much of the literature as of 1997

  > "ignore[s] the effects on the [analogue] computing process of *imprecision* and *noise*, two of the most pervasive practical problems in analog computation".

  Eleven years on, [35] notes that

  > "this evaluation remains largely accurate".

- A demonstration ([116]) that random-access machines, when augmented with infinite-precision, real-number manipulations, can solve in polynomial time and space any problem in NP. This indicates, additionally to Sect. 3.1.1, that unconventional resources such as precision should without doubt be considered.

- Introduction ([69]) of '*measurability*' as an analogue-computer counterpart to the Turing machine's computability, and exploration of the consequences of computability or otherwise of physical theories' mensurable values.

- An observation ([35]) that,

  > "[a]lthough it has been shown that some continuous time models exhibit super Turing power, these results rely on the use of an infinite amount of resources such as time, space, precision, or energy. In general, it is believed that 'reasonable' continuous time models cannot compute beyond Turing machines".

Aside. We make a final comment in this section. One may question whether *chaotic mappings* can be exploited so as to increase available precision, and hence to reduce precision complexity. Two close values (that we wish to distinguish, though may not have available sufficient precision) may, under a chaotic mapping, have distant and easily distinguishable images; the difficulty is then transferred from precise distinction of values to reversal of typically unpredictable chaotic functions. Full investigation of this idea is deferred to future work.

3.4 Formalizing Resource

In this section, we formalize the notion of 'commodity resource' (recall the discussion of Sects. 3.2.1 and 3.2.4). Where convenient in the remainder of this dissertation, we use the briefer term 'resource'.

3.4.1 First Steps

We model a *resource* (usually denoted by an upper-case letter A, B, etc.) as a function that depends upon the choice of *computational system* (shown as a subscript to the function, which subscript may, when understood, be suppressed) and that maps each *input value* to the corresponding amount[80] of the resource consumed by the system in processing the input value (up to the point of providing a corresponding output value).

In fact, we stipulate that resources map to *natural-number*[81] values (or to ∞)[82], which can, therefore, be thought of as the corresponding numbers of *units* of resource consumed. Should our resource instead lend 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.

Hence, 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* since we stipulate that resource A have codomain $\mathbb{N} \cup \{\infty\}$—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;
- $S_\Phi(x)$, the number of units of *space* occupied (e.g., the number of tape cells used if Φ is a Turing machine);

[80] 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).

[81] Recall that, in the present context and unlike in the previous chapter, zero is a natural number: $\mathbb{N} := \{0, 1, 2, \ldots\}$.

[82] We use this fact in the proof of Theorem 22, for example.

- $P_\Phi(x)$, the number of units of *precision*, as defined in Sect. 3.3.3, used by the system; and
- $\mathbf{0}_\Phi(x)$, which is equal to 0 for every computer Φ and input value x—this *null resource* is introduced (in Definition 17) for technical reasons.

Remark 27. Clearly, we have not fully *defined* resource, but have instead mentioned some necessary (but by no means sufficient) properties. We desire for our purposes further properties, some of which we now describe.

Blum's Axioms.

In standard complexity theory, i.e., complexity theory as it applies to Turing machines, Blum's axioms (introduced in [31]) 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 [31]—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 at precisely those inputs at which the computation being measured (and during which the resource is consumed) is itself defined, and
2. that it is a (Turing-) decidable problem to determine whether a given (purported) value is indeed the measure of resource corresponding to a given input.

Both axioms hold, for example, for the Turing-machine resource of *time*: (1) the number of time steps elapsed during a computation is a well-defined, finite natural number if and only if the computation halts; and (2) given a Turing machine, an input value and a purported number n of time steps, we can decide—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 a good starting point for our restricting suitably the notion of resource in the context of unconventional computation. Axiom 1 is automatically satisfied (this is implicit in the phrase "corresponding amount" in the description of resource in Sect. 3.4.1, wherein the implication is that an undefined computation, and only an undefined computation, gives rise to an undefined/infinite amount of resource), and—for our measures of resource to be of any practical value—the desirability of axiom 2 is clear.[83] Further, our adoption of the axioms gives us many standard results from [31]; e.g., (Theorem 3 of [31]) for resources A and B, and for dummy variable x standing for an input value, there exists a computable function g such that $g(x, A(x)) \geq B(x)$ and $g(x, B(x)) \geq A(x)$ with only finitely many exceptions x.

Further Restriction.

We stipulate that resources satisfy Blum's axioms. However, necessary as we deem them to be, the axioms are not *sufficient*; we see in Sect. 4.5.1 that a

[83] In particular, we do not consider here *non-deterministic* resources (even though computers themselves may be non-deterministic), which would require that axiom 2 not hold.

notion of resource constrained by Blum's axioms alone leads to undesirable and deceptive complexity behaviour—whereas we describe below (see Sect. 4.2) tools 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.

Accordingly, we further restrict[84] the notion of resource in Sect. 4.5.2, after which (in Sect. 4.5.4) we summarize for convenience of reference our modified notion.

3.4.2 Null Resource

It is convenient to define a *null resource* that is consumed (in positive quantity) by no computer.

Definition 17. Let the *null resource*, denoted by $\mathbf{0}$, be the resource that is consumed by no computer in that $\mathbf{0}_\Phi(x) = 0$ for all computers Φ and input values x.

The corresponding complexity function $\mathbf{0}^*_\Phi$ (see Definition 18) is, therefore, constantly equal to zero for all computers Φ.

3.4.3 From Resource Comes Complexity

Even without fully defining resource, it is possible to define complexity functions in terms of resources. We do this now.

With each resource is associated a *complexity function*. Given a computing system, one may ask how a specific resource scales: we (as complexity theorists) may be interested not in the resource required by the computer in processing *one specific input value* (i.e., in some '$A(x)$'), but in the resource required *as a function of the input value's size*[85]—this is the *complexity function* corresponding to the resource. Specifically, we make the following definition.

Definition 18. Let Φ be a computer with set X_Φ of possible input values, let σ be a size function (see Sect. 3.4.3 *Size Functions*), and let A be a resource. The *complexity function* $A^*_{\Phi,\sigma}\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$, corresponding to resource A, is given by

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

('Φ's and/or 'σ's may be suppressed when understood).

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

Note that our previous, ad hoc use of 'T^*' and 'S^*' (in Chaps. 2 and 3), and of 'P^*' (in Chap. 3) is in accordance with Definition 18.

[84] Specifically the restriction stipulates that resources be *normal* (which term is defined in Definition 27). 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. This prevents the exaggeration of the (complexity-theoretic) importance of resources by, for example, applying a quickly-growing monotone function; see Sect. 4.5.1.

[85] For a discussion of the way in which a value's size may be defined, see Sect. 3.4.3 *Size Functions*.

We reiterate that we have not *defined* resource. The description given above 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.

Remark 28. Much as we should like to ascertain the complexity of *problems*—typically mathematical tasks such as factorization or the Travelling Salesperson Problem that are of natural interest—, it is usually possible to measure the complexity only of *methods that solve the problems*—computers, programs, algorithms and so forth—; this method-complexity bounds from above the sought problem-complexity (see also Sect. 4.1.1 *Complexity: Problems versus Solution Methods*). Accordingly, complexity functions as we define them here are properties of computing systems.

Remark 29. Note that, regardless of the choice of resource, the corresponding measure of complexity (and, indeed, complexity classes—see Sect. 4.3) are defined in terms of the resource in the same way. The motivation is that this allows fair comparison of different measures of complexity (for more about which see Chap. 4); a byproduct is that our complexity framework is easily extensible: the introduction of a new resource leads automatically to a new measure of complexity and corresponding complexity classes.

Size Functions.

Since complexity functions are defined (in Definition 18) so as to take as their argument an input value's *size*, this term needs to be understood.

Definition 19. A *size function* (which we often denote by 'σ') maps each input value to a member of $\mathbb{R}^+$, which member we call the input value's *size*.

Example 2. For example, if our input value is a *natural number* n expressed in base b (typically, b will be 2 or 10), then we can take as its size

- the number $\lfloor \log_b n \rfloor + 1$ of digits, excluding leading zeros (or, as is sufficient for virtually all complexity-theoretic purposes, the often-convenient approximation $\log_b n$ to this number; we can further simplify this to $\log n$, since the choice of base equates merely to multiplication by a constant, which detail is ignored by $\mathcal{O}$-notation—see Footnote 5 of Chap. 2).

If, instead, our input value is a *real number* x, then, depending upon context (how x is encoded/stored[86], how it is to be used, etc.), we can take as its size

- the natural-number size (e.g., in accordance with the list immediately above) of $\lfloor x \rfloor$;
- the combined number of digits and decimal places of x (supposing presentation to be in base notation with terminating decimal expansion);
- the Kolmogorov (algorithmic) complexity of x (see also Sect. 6.1.2); or

[86] We recall for its innovation the suggestion in [101] that real numbers may be represented (e.g., in abstract descriptions of computations) using ideas based upon Dedekind cuts.

- the combined number of digits of the numerator and denominator of x (supposing presentation to be as a fully cancelled rational expression[87]).

As the size of, say, a conjunctive-normal-form formula of Boolean logic, we may take

- the number of free variables in the formula; or
- the number of clauses (i.e., conjuncts) of the formula.

Where the particular choice of size function σ is unimportant, we sometimes write '$|x|$' for '$\sigma(x)$'.

We defer further detail on size functions to the standard complexity-theory literature, and, in the particular case of real-number input values, to [30]. Here as in traditional complexity theory, these size functions are not our focus; we tacitly assume that such functions are chosen 'sensibly' (were 'sense' here depends upon context), and focus instead upon the higher-level notions (complexity function, complexity class, etc.) that we define in terms of size functions.

Resource as a Lower Bound.

We comment in passing that specific values $A_\Phi(x)$ of resources and $A^*_{\Phi,\sigma}(n)$ of complexity functions are viewed as *lower bounds* on what is needed for a computation to succeed: we tacitly make the prima facie reasonable assumption that computation can proceed when more resource than is necessary is available.[88]

A situation where this assumption fails is when some resource A has an *upper* bound for computation to succeed, when there is such a thing as 'too much of A'. In this case, one may introduce a new resource, '$-A$', bounded from *below* as we assume here; this situation is accommodated, then, even though we take values of resources and complexity functions to be lower bounds.

Our default use of resource (i.e., as a lower bound), then, is to make statements of the form 'a computation can proceed if *at the fewest* X units of the resource are available'. In the previous paragraph, we see that we may also apply upper bounds: 'a computation can proceed if *at most* X units are available' (to bound A *above* by X, we simply bound $-A$ *below* by $-X$). This allows specification of a (two-ended) range of values required of resource availability; as we discuss now, however, there may be more subtle constraints on computational resources.

Note that, in a quantum (and, hence, quantum-computing) context, additional 'possibilities' (for example, potential routes taken by photons in Young's double-slit experiment [136]) may interfere with *and cancel out* existing ones (see [105] and Fig. 3.9); the existence of such phenomena should be heeded, then, when selecting resources for consideration, so that provision of extra resource cannot, all else being equal, preclude a computation—we wish to be able to

[87] Alternatively, x may be given as a rational *approximation*, not necessarily fully cancelled, where the denominator d indicates the *resolution* to which x is given, in that x is the true value rounded to the nearest $\frac{1}{d}$; e.g., rather than cancelling $\frac{50}{100}$ to $\frac{1}{2}$, it is left so as to indicate that the true value is in the interval $\left[\frac{49.5}{100}, \frac{50.5}{100}\right)$.

[88] A desirable byproduct of this assumption is that our unconventional-complexity definitions are in some respects analogous to their traditional counterparts: certainly no Turing machine M is hindered by an abundance of time or space, in that the class of computations possible given t time steps and s tape cells is $\subseteq$-monotonic in t and (separately) in s.

take suprema, and otherwise to consider our lower-bound resources/complexity functions, with impunity. Note, reassuringly, that the problematic 'resource' described here is not really in keeping with out notion of *commodity* resource: the availability or otherwise to a photon of a certain route is a feature of the (problem-instance-independent) *structure* of the system; our stipulation that resources (and complexity functions) be lower bounds is not in conflict with the photon-path situation described.

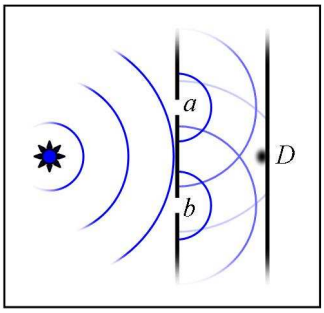

Figure 3.9: Less is more: if both slit a and slit b are open, then light is cancelled out at (dark) point D; with only one slit open, D becomes lit.

Remark 30. For clarity, we stress the sense in which values of resources and complexity functions are lower bounds. A computation *with fixed input value* may proceed with at least these (resource/complexity-function) bounds' allocation of computational 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.

In particular, the definition of $A^*_{\Phi,\sigma}$ in terms of the *supremum* of a set of amounts of resource reflects the fact that, whilst there may be several different amounts $A_\Phi(x)$ of resource corresponding to various input values x of the same size n, we are interested in finding an amount $A^*_{\Phi,\sigma}(n)$ of resource sufficient *for any* input value of size n.

3.5 Model-Independent Resources

An important question here is

($\star$) *what methods can be used to identify the suitable/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/what can be said about the sought resources?'—or for specific models; we consider both levels of generality (see Sects. 3.5 and 3.6 respectively).

Aside. The temptation is to say that 'everything'—properly, every function mapping the value of some feature of a computational system into the set $\mathbb{N} \cup \{\infty\}$—is (or at least can be) a resource, in the hope that most such things can

be disregarded as having no dependency on the input value (which is the case, for example, when such a 'resource' is consumed in zero quantity). However, we wish, at the very least, for statements along the lines of 'the total amount of resource consumed by computer Φ given input value x is k' to be decidable, which entails our being more decerning in our choice of resources. We do not focus on this issue in the present project, but do make restrictions of the notion of resource (see Sect. 4.5.4) that are sufficient for our purposes.

Some clarification is needed on what it means for a resource to be 'suitable/relevant'. An intuitive understanding of this concept can be gleaned from the analogue factorization examples of Chap. 2, which we have demonstrated to have time and space complexity (each polynomial) that fails to capture the systems' actual, *exponential* complexity; the unconventional resource of *precision* harbours the true complexity here, and, hence, is seen to be *relevant* (unlike time and space). We formalize this 'relevance' with the notion of *dominance*; see Chap. 4.

Remark 31. Note that, whereas the phrasing of our question ($\star$) suggests that resources' suitability/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 have, say, time but not precision as a relevant resource, whilst, to another of the same model, precision but not time may be relevant (we show the 'difficult half' of this claim, for the specific case of analogue computers, in Chap. 2). 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 the notion of *dominance*, which is defined and described below).

3.5.1 Specific Examples

We discuss first some resources that apply to all (or virtually all) computational models.

Precision.

We discuss the resource of *precision*, which above furnishes us with examples and deserves detailed treatment, in Sect. 3.3. This resource is not relevant to all computational models (as is witnessed by the notable example of the Turing machine—recall Theorem 1), but, as suggested by the discussion of Sect. 3.3, its consideration is certainly beneficial when working with paradigms prone to input/output noise; the prevalence of such paradigms, especially when one restricts consideration to computers *as physically implemented*, earns precision its place in Sect. 3.5 rather than Sect. 3.6.

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—see the discussion of Sect. 3.7.1. The

problem, as we intimate above, is that it is sometimes assumed that this limited interpretation of resource is adequate, even in the unconventional case.

Whilst we see above (particularly in Chap. 2) 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; in fact, they are universally applicable. Furthermore, it is clear[89] for virtually all computing paradigms how to generalize these two resources from the Turing-machine case to the wider class of physical computers.[90]

Definition 20.

- *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;[91]
- *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).

Remark 32. Our choice of specific units (s and m^3) is arbitrary and is included only so that the resources of time and space are well defined. The complexity-theoretic (and in particular asymptotic-notation) use of this definition is such that this choice is equivalent to any other.

Remark 33. Note that our generalizations of time and space encompass all relevant parts of a computational process, including measurement of output values. This is in accordance with the view—expressed in [8], for example—that complexity measures should account for measurement processes: for example, if an output value is taken from a balance scale that tips arbitrarily slowly as the compared masses approach equality, then the duration of the tipping process should form part of the computation's time, which is indeed so with our definition.

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 as a non-commodity resource—see Sect. 3.2.1 *Manufacturing Costs*), there may be a cost in *running* it (over and above energy costs, which we discuss in Sect. 3.6 below). For example, if memory is implemented in such a way that each write operation (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 (in particular for physically implemented computational paradigms) to the resource of *material cost*.

[89] Clear, at least, given suitable notions of normalization, etc.; see Sect. 4.5.

[90] Recall that we take above the first step of the generalization: in Sect. 3.3.3 *Time Complexity; Space Complexity*, we extend the conventional-computer notions of time and space complexity (as presented, for example, in Definitions 2.5 and 2.6 of [103]) to PSMs. We now take this extension to its natural conclusion, by generalizing time and space complexity to all physical computational models.

[91] Here we implicitly assume that computations are not affected by relativistic effects, which assumption is clearly unsafe in the context of relativistic computation; for this model, then, our definition of time needs modification—see Sect. 3.6.1 *Relativistic Computers*.

Thermodynamic Cost.

Strictly speaking, this resource is not applicable to computers from *all* models (failing, in particular, for those from abstract, mathematical paradigms), 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 an erasure of information, an increase in entropy and, hence, a thermodynamic cost; this concept was introduced by Landauer in [92] and developed by Bennett and Vitányi (who survey the notion in [15,129] respectively) amongst others. Reference [137]—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[92] (see Sect. 6.1.2). Note that, at least from the perspective of [137], 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—what we have is a very much theoretical lower bound on the consumption of 'thermodynamic resource'.

Aside. We recall from [53] an approach to computational resource that leads to other information-theoretic resources. In practice, *randomness* (roughly speaking this is non-determinism viewed as a commodity resource) comes for free—one needs only to toss a coin (whilst true for virtually all computational, function-evaluation purposes, this is not, however, the case in the context of *cryptography*, where the slightest deviation from true randomness is open to exploitation); significant cost is incurred not by use of this randomness, then, but rather by use of *correlations* and *structure*. These features roughly equate to information storage and processing, from which observation it is understandable that information-theoretic resources such as entropy are involved.

Aside. We recall (from [94]) in passing also the notion of *thermodynamic depth*, which provides a measure of the amount of information lost in formation of an object (a computer being the example of interest in the present context), rather than in a computational process. This is not, then, a commodity resource, but rather a manufacturing cost closely related to the commodity resource of thermodynamic cost; see Sect. 3.2.1 *Manufacturing Costs*.

3.5.2 Generic Criteria

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; in fact, we discuss two such elsewhere: *Blum's axioms* in Sect. 3.4.1 *Blum's Axioms* and *normalization* in Sect. 4.5. We defer details to these sections.

[92] Note that, in the general case, algorithmic complexity is uncomputable: we fail not only to find a smallest program that generates a given string, but also to find even this program's size.

3.5.3 Two Approaches

> "In theory, there is no difference between theory and practice.
> But, in practice, there is."
>
> —*Jan van de Snepscheut.*

We comment in passing that many of the approaches to identification of suitable/relevant resources fall into the broad categories of 'practical' and 'theoretical', as we now describe (these approaches are akin to the discussion of Sects. 3.5.1, 3.6.1 and 3.6.2 in the former case and Sect. 3.5.2 in the latter).

- Practical approach. With a specific computing device or paradigm 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, since the result of a computation by an analogue system 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, we are prompted 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.[93]
- Theoretical approach. We may question, regardless of computing system/paradigm, what can and should be admitted as a resource. As we comment above, Blum's axioms—see Sect. 3.4.1 *Blum's Axioms*—offer an effective starting point, and normalization—see Sect. 4.5—a suitable continuation; see also Sect. 4.5.4 for a summary of our restrictions to the abstract notion of resource.

3.6 Model-Dependent Resources

3.6.1 Resources for Non-Quantum Computers

We consider now specific (illustrative rather than exhaustive) models of computation, and ask which resources are likely to be relevant for instances thereof. Due to the importance and timeliness of *quantum* computers, and to the sheer number of distinct quantum-computing paradigms, we treat quantum computers (in Sect. 3.6.2) separately from non-quantum computers (in this section).

In light of Remark 31, we here seek to identify for a given computational paradigm *candidate* resources that may be of interest for *some* instances of the paradigm; in particular, we do not (and in most cases cannot, due to the falsehood of the proposition) provide rigorous proof that a resource is of interest for *all* instances of a model of computation.

Actual, Physical Implementations of Turing Machines.

We comment in Sect. 3.7.1 that, for the abstract, unimplemented, Turing-machine model itself, the resources of time and space (and variants thereof) are

[93] This, recall, is notably the case with the analogue factorization systems of Chap. 2, for which precision transpires to be paramount.

sufficient for complexity-theoretic purposes. The resource of *precision*, then, is not a direct concern for Turing machines:

> "What 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" [128].

What, though, of *implementations* of Turing machines? Both the *alphabet size* and *number of states* of a physical realization of a Turing machine relate to precision in that distinguishing a greater number of distinct symbols entails smaller differences therebetween,[94] resulting in smaller differences (in voltage or similar) between their respective *real-world implementations*.[95]

Nonetheless, the alphabet-size issue is not problematic: 'meta-symbols' consisting of several symbols can be used instead of (distinct) individual symbols; similarly, the states may be encoded via such 'place notation' (for example, states' indices may be recorded in a multi-column, place-notation register rather than as a single symbol). 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 (that may or may not have been manufactured at the time of the computation's commencing); we therefore consider space as a commodity, rather than a manufacturing/material, cost.

This place-notation approach suggests that consideration of time and space alone may for many purposes suffice, and (unsurprisingly) Turing [127] 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 for precision we should certainly account.

So, we consider for this model the resources of *time* and *space* (of which the 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.

(One resource not considered thus far is the *energy* consumption of the computer—the electrical cost, say, of its operation. This, however, is a constant multiple of time—to power a computer has a *unit* energy cost per second,[96] and,

[94] We tacitly assume, perfectly reasonably, that there exists some bounded space from which symbols are drawn.

[95] Strictly speaking, symbol and state numbers for a given Turing machine 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. This contrivance does not 'cheat a new resource into being', but rather ensures that (not-necessarily-commodity) resources—namely, numbers of symbols/states—that we should like to discuss can be expressed as commodity resources; this, by the discussion of Sect. 3.2.2, is harmless.

[96] Clearly, periods of more intensive computation may require more energy in order to power the processor, but we may (and do) assume 'ballistic computation', whereby the computer is

hence, energy becomes redundant as long as we consider time: they measure, in effect, the same quantity.)

Aside. Though the treatment in [42] of the main theme (and specifically the 'proof' of the paper's title, $\mathsf{P} = \mathsf{NP}$) leaves a lot to be desired, the paper does at least prompt an interesting aside: one may ask which of the (countably) infinitely many abstract Turing machines can be physically implemented. We suggest (given the above comments concerning the differences between an abstract Turing machine and a physical instantiation thereof, and given the typical unboundedness of required space/time as input values become larger) that the question is more naturally posed and more readily answered in the form 'which Turing-machine/*input-value* pairs can be physically realized?' This, we suggest, is ultimately a question about consumption of commodity resources.

In summary, the resources relevant to ***actual, physical implementations of Turing machines*** *are* ***time***, ***space*** *and* ***precision***.

Aside. We argue here that consideration of this resource set (i.e., $\{T, S, P\}$) is sufficient, and that consideration of any proper subset thereof is insufficient, for insightful analysis of the complexity of real-world implementations of Turing machines. We comment as an aside that there may exist different choices of resource set with this property (though conjecture at least that all such sets have the same cardinality), not least in light of the trade-offs that exist between resources (see Sect. 3.8.1). It is hoped, however, that our treatment of resource, and in particular the restrictions—normality, etc.—to the notion, render the complexity behaviour of a computer 'essentially' (i.e., '$\mathcal{O}$-') identical, regardless of the choice of exhaustive resource set.

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 systems of Chap. 2, the true (exponential) complexity of which these two standard resources fail to capture. One additional relevant resource, as we see above, is *precision* (this is evident from consideration not only of the factorization systems but also of the greatest common divisor system described in Sect. 3.3.2 *Greatest Common Divisor*, the wedge-detection cannon system of [8, 9, 12] mentioned in Sect. 3.3.4, the Differential Analyzer of [43] discussed in Example 3, etc.); and, as we see below, there are yet other resources germane to this computational model.

It seems intuitively clear (or, at the very least, plausible[97]) that one should consider also the *energy* required to drive the computer; this is evoked particularly strongly by the example of a kinematic computer with its (eponymous) moving machinery. Energy was considered only parenthetically in the case above

given an input value, is set in operation, and computes continually (and at maximal processor capacity) until an output value is found. Note that this assumption does not restrict the expressivity of our notion of computation: the alternative to ballistic computation—namely, interactive computation—can be modelled as a series of ballistic sub-computations separated by user interactions. (The term 'ballistic computation' was brought to the author's attention by—and is, we believe, due to—Susan Stepney.)

[97] Recall that plausibility is sufficient: we are identifying candidate resources that may be relevant to some instances of the model, rather than proving relevance for all instances.

(namely, physical 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 vice versa) and therefore redundant from a complexity-theoretic perspective (provided that we have not neglected to consider time); this is in contrast with the present model—the analogue/kinematic computer—, however, since energy consumption may vary from time step to time step (much like the space consumption of a Turing machine).

Example 3. We discuss now an important example of analogue computers: the Differential Analyzer of [43].

This system solves, via analogue means, differential and integral equations. In practice, the manufacture of the system's components a priori determines the system's *precision*, which in turn constrains the inputs that can successfully be processed[98]; precision, then, is seen as a manufacturing resource, the availability of which limits the allowable inputs (compare this with the approach to precision in [128]—see Example 4), though clearly has an equivalent commodity-resource formulation (recall Sect. 3.2.2). Formulation aside, the key observation is that precision is a resource that warrants consideration, for the Differential Analyzer specifically and the analogue-computing paradigm generally.

Regarding the resource of *time*, preparation of the system takes a matter of hours (though not tens of hours), and solutions are presented after about ten minutes' processing; one expects that, were input intricacy unimpeded by the precision concerns of the previous paragraph, the required processing—and possibly also preparation—time would increase along with the input size. Further, there are many engineering technicalities that contribute to the system's run-time—for example, the system features flywheels that (at a slight time cost) damp unwanted oscillations in the integrator gadgets, and uses additional gear revolutions proportionally to lessen backlash effects (again, at a time cost)—; though this contribution is unproblematic with the 'small' input instances described above, it is not clear how it would scale asymptotically.

Regarding *space*, the intricacy of the input equations is limited not only by the components' precision, but also by their number; this is akin to the commodity resource of space, but also has closely associated non-commodity (material, manufacturing, etc.) costs.

This example bears out that the relevant resources for this model include time, space and precision. Reference [36] discusses the computational power of Shannon's *General Purpose Analog Computer*—which is a mathematical abstraction, introduced in [117], of the Differential Analyzer's computational model—, and identifies no further resources than those discussed in this example.

Example 4. In [128], the resource of precision in the context of analogue computation is dealt with by acknowledging that such a 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 rendering precision a non-commodity resource. Various commodity resources are also considered; these

[98] Specifically, these inputs are ordinary differential equations of order up to six with "any amount of complexity within reason" [43]; these limitations stem from an imprecision of approximately one part per thousand for each component, and a greater resultant overall imprecision.

include time, space, energy and mass (which last, due to the bounded density of the system's constituent parts, is subsumed by the resource of space—cf. Sect. 3.6.1 *Chemical/DNA Computers*).

The paper offers further indication that a complexity-theoretic formalization of the notion of precision is both natural and necessary:

> "if "infinitely accurate" analog devices could be built, then they could be used to solve 3-SAT arbitrarily fast".

Example 5. In [67], we see logic gates implemented using elastic, billiard-ball collisions. Reference [123] builds upon the work by instantiating gates using repelling blocks (of which some are fixed) so as to implement logical operations; these systems are modelled as two-dimensional cellular automata, with a tacit assumption therefore being that the blocks are precisely aligned with a grid—in practice, of course, such a system would be susceptible to a slight imprecision in the blocks' positions, potentially to the extent of the system's functioning incorrectly, confirming once more that precision warrants consideration when dealing with analogue/kinematic computers.

In summary, the resources relevant to ***analogue/kinematic computers*** *are* ***time***, ***space***, ***precision*** *and* ***energy***.

Relativistic Computers.

The broad idea of relativistic computation (of which the detailed physics is beyond the scope of the present project) 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 it exists in the context of other models.

Secondly, note that, for present purposes, we adopt a fairly practical stance when considering models of computation: when identifying resources, we are careful to distinguish between (abstract) Turing machines and their (physical) implementations, for instance; further, we consider practical, 'physical' issues such as achievable input/output precision. In this context, then, it seems reasonable to exclude relativistic computers (at least as a form of hypercomputer) on the grounds of their energy consumption, for example: 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 in an attempt

to bypass *time* restrictions, but other resources' constraints (energy, durability of the physical machine, etc.) are still present.[99] We see in another guise our fundamental 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 or space is still an issue.

Remark 34. What we describe here as the paradigm of relativistic computation is, in fact, a *family* of paradigms. The relativistic aspect is a harness that contrives to buy time for an auxiliary computer (that which is sent through the wormhole or similar) of unspecified paradigm; for each physically implemented model of computation, then, one can envisage its relativistic-computational use. When we write above of "energy, durability of the physical machine, etc.", we mean the commodity running costs incurred and defined by the model of the auxiliary computer (whatever these costs may be).

(The commodity resources of user- and computer-time, energy, etc. 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. 3.2.1 *Manufacturing Costs*.)

In summary, the resources relevant to ***relativistic computers*** *are* ***user-time***, ***space*** *and the* ***auxiliary computer's running costs***.

Optical Computers.

Reference [133] introduces an optical system that computes via image manipulation; the seven (commodity) resources considered in that paper are

- time,
- number of images,
- spatial resolution (i.e., the number of pixels actually needed for the computation to succeed),
- amplitude resolution,
- phase resolution,
- dynamic range (i.e., the maximal amplitude encountered during a computation), and
- frequency of illumination (i.e., the minimal optical frequency for the computation to succeed).

[99] As a less 'physics-dependent' illustration of this point, we recall from discussion with Cristian Calude the *accelerated Turing machine*, of which *space* consumption may be prohibitive—infinite, even—, the bypassing of *time* concerns notwithstanding; see [45].

Note that the *number of images* is a measure (albeit taken in different units) of what we have previously defined as the resource of *space*; that *spatial*, *amplitude* and *phase resolutions*, and *frequency of illumination* are closely akin to our *precision*; and that *dynamic range* is akin to *space* (which can be seen upon consideration of a simulation of such an optical system whereby amplitude is stored as an integer in a register—the greater the dynamic range, the more digits are required). These seven resources, then, are akin to forms of *time*, *space* and *precision*, of which each is relevant for the optical computers of [133].

(The relevance of precision to optical computers is evident, also, in our discussion of the ray-tracing problem—see Sect. 5.2.)

We suggest further 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—consider, for example, an electromagnetic-wave implementation of the gcd system of Sect. 3.3.2 *Greatest Common Divisor*.

In summary, the resources relevant to ***optical computers*** *are* ***time***, ***space***, ***precision*** *and* ***energy***.

Chemical/DNA Computers.

One commodity resource that is particularly relevant to chemical computers is *mass*. We recall from [3] 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 is acceptable. However, as is pointed out in [74], the *mass* of DNA required by the method in processing non-trivial problem instances is greater than the mass of earth![100] 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.[101]

As [74] concludes,

> "[t]his leaves us with the difficult task of understanding what computations can be performed below the exponential computational resource bounds imposed by nature."

Aside. Note that, due to the bounded density of chemical-computing apparatus —including DNA strands themselves—, mass is bounded by a constant multiple of space, and so the resource of mass seemingly 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.

[100] Hartmanis [74] 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 (nor, hence, a simple, linear relation between weight and mass).

[101] 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 Travelling Salesperson Problem system and similar stems essentially from the presence of exponentially many DNA strands simultaneously testing one potential solution each. See, however, the discussion of parallelism in Sect. 3.7.1.

Remark 35. We remark that arbitrary instances of this paradigm are essentially probabilistic: there is always a positive probability—albeit typically small and, furthermore, diminishable via repetition—that the output value returned by a non-trivial chemical computation is incorrect.[102] A problem is not generally placed, then, into a standard complexity class (other than those defined in terms of probabilistic solutions) by virtue of its being solved by a chemical system. For instance, a chemical system that solves the problem of protein-folding (and, by definition of the problem, there are *natural* such systems[103]), even with polynomially scaling resources (not only time and space, but also mass, energy, etc.), does not contradict the NP-hardness of the problem.

Example 6. As an example of other resources that arise in the context of chemical/DNA computation, we recall from [46] that DNA gates can be implemented such that computation is performed via chemical reaction; DNA strands encoding input values react to produce strands encoding output values, along with waste.

The process can be contrived such that the waste from a gate does not interfere with that gate's reaction, though a problem that remains unsolved (at time of [46]) is that waste from one gate may interfere with the functioning of other gates. This suggests various potential complexity measures: the number of concurrent invocations of gates, the spatial/temporal proximity of invocations of gates, etc. Spatial proximity in particular seems reminiscent of our resource precision (and, for the purposes of summarizing this paradigm's relevant resources, we class this example's resources as such).

In summary, the resources relevant to ***chemical/DNA computers*** *are* ***time***, ***space***, ***precision***, ***energy*** *and* ***mass***.

3.6.2 Resources for Quantum Computers

We turn now to consideration of the resources present in quantum computation, beginning with *circuit-model* quantum computation. We hardly need comment that insightful study of the complexity of quantum computers is crucially important and timely: the benefits of formalization, in computational-complexity terms, of the technological difficulties (which stem ultimately from the presence of unconventional resources) faced when implementing working quantum computers are clear. By introducing and considering new resources, specifically ones similar to precision, one may, we suggest, better encapsulate the true complexity of quantum computers; this complexity, after all, is fundamentally connected to our limited ability to take precise measurements from the system (see also Sect. 5.4).

As in Sect. 3.6.1, Remark 31 prompts us to seek to identify for a given computational paradigm *candidate* resources that may be of interest for *some* instances of the paradigm; we need rigorously show neither that such resources

[102] This claim is corroborated in [104].

[103] Though natural systems seem to fold proteins relatively efficiently from a *commodity-resource* point of view, consider the (non-commodity) manufacturing costs—see Sect. 3.2.1 *Manufacturing Costs*—incurred in evolving such systems! This suggests a 'meta-trade-off' not between resources, but between *interpretations* of resource (in this case, commodity and manufacturing); the intuition here is clear—time can be invested in designing better computers—, though formal investigation is beyond the scope of the present project.

are of interest for *all* instances of a model, nor that there exist no other such resources.

Aside. We mention in passing that a desirable feature of computational models (quantum or otherwise) is their possession of high-level 'gadgets' that can be combined in a fairly intuitive way to achieve chosen computational aims. This is the case, for example, with the random-access model (as implemented by suitable programming languages), since there exist high-level languages featuring commands that can be used in well-understood ways to produce arbitrary (computable) effects without the programmer's having to resort to bit-level manipulations; it is also true, to an extent, of reaction-diffusion systems: [70] introduces relatively high-level gadgets (read-write memory cells, switchable channels, etc.) for computing with pulses of excitation in such systems.

However, the same cannot be said for quantum-computing systems. Quantum algorithms are derived in an ad hoc fashion via low-level quantum-bit/gate manipulations, direct exploitation of physical phenomena, etc., rather than being 'written' using a high-level library of 'commands'. We recall Bob Coecke's recognition (expressed in, e.g., [50]) that high-level tools for producing quantum algorithms are sadly lacking.

Circuit-Model Quantum Computers.

An arbitrary Turing-machine (or similar, standard-model) computation can, by definition of 'complete', be expressed as a conversion of input to output via operations taken exclusively from a complete set of what are deemed to be 'atomic' operations (whereby tape cell, machine state and head position are updated, for instance). For a given input value, the number of such operations performed during this conversion is an accurate measure (or, depending on viewpoint, a definition) of run-time.[104]

Similarly, an arbitrary *circuit-model quantum* computation can be expressed as

- the preparation of several quantum bits (which encode the input value), followed by
- a sequence of applications (which constitutes the computation's processing stage) to subsets of these quantum bits of 'atomic' unitary operations taken from a complete set, followed by
- measurement of the system (from which is obtained the output value)

(see [99]). As in the classical, Turing-machine case, an enumeration of the invocations of these atomic operations gives a measure of the system's complexity; indeed, this is the basis of an existing definition of complexity of circuit-model quantum computing devices (see Sect. 4.5 of [99]). Also as in the classical case, however, this measure essentially captures the system's *run-time*, which is not, we suggest, a particularly insightful measure for quantum computers:[105]

[104] The more accurately the complete set reflects the environment (e.g., the chip-set, and, specifically, the atomic instructions thereof) in which the system is implemented, the more accurate the measure.

[105] That a quantum system may be significantly more time-efficient than a Turing-machine counterpart is certainly of great interest, though if this is at the cost of efficiency in terms of another (possibly unconventional) resource, then this latter fact is also relevant and telling.

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 (with run-time being an underestimate due to parallelism's not being taken into account) nor the 'difficulty' in using the system (which arises chiefly during measurement rather than as a result of the user's having to wait for lengthy computations to finish)—a reflection, that is, not of the *complexity* of the system (see Sect. 3.2.3).

The broadened notions of resource and complexity proposed in this project, and particularly the resource of precision, seem to encapsulate better the nature of the true complexity of circuit-model quantum systems; this complexity arises, after all, because of our limited ability to take precise measurements from the system. More concretely, we note from [83] that the SAT problem does not succumb in an obvious way to a circuit-model approach since *uniquely satisfiable* input formulae would be encoded as states exponentially close to those encoding *unsatisfiable* formulae. Accordingly, we suggest that a resource important and relevant to this paradigm (along with the ubiquitous *time* and *space*) is *precision* (not least because of consideration of *proximity of quantum states*, as in the SAT instance above).

In summary, the resources relevant to ***circuit-model quantum computers*** *are* ***time****,* ***space*** *and* ***precision****.*

Adiabatic Quantum Computers.

We consider now the *adiabatic* quantum model (wherein output values are encoded in the final ground state of an evolving system, with the evolution proceeding from an easily achievable initial ground state sufficiently slowly that no higher energy state is reached, so that the 'output-value' final ground state is indeed encountered). Standard expositions of the paradigm (such as [64]) 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.

We defer more detailed discussion of this paradigm to Sect. 5.3 (wherein we focus on controversial claims of the paradigm's being able to solve an undecidable problem), but suggest here that the relevant resources are *time*, *space*, *precision* and *energy*.

In summary, the resources relevant to ***adiabatic quantum computers*** *are* ***time****,* ***space****,* ***precision*** *and* ***energy****.*

Measurement-Based Quantum Computers.

The measurement-based quantum-computing paradigm, described in [109], sees a computation take the form of several measurements (which can, in principle, be performed simultaneously), each of an individual quantum bit, starting from an initial, large, entangled (cluster) state. As this description sug-

gests, the complexity resources relevant to such a computation are, potentially, markedly different from those used in the circuit and other quantum models (and even more so from non-quantum paradigms). In fact, however, a chief difference between this and other models—specifically, this difference is the reliance of measurement-based quantum computers upon availability of the initial cluster state—is an issue of *manufacturing* (i.e., non-commodity) resource; see Sect. 3.2.1 *Manufacturing Costs*. Commodity resources, we claim, are as circuit-model quantum computers.

In summary, the resources relevant to ***measurement-based quantum computers*** *are* ***time***, ***space*** *and* ***precision***.

Continuous-Variable Quantum Computers.

We suggest that the resource of *precision* is (along with the ever-present *time* and *space*) of relevance to the continuous-variable quantum paradigm, which is described in [39] (we do not claim that there are not other relevant resources, but focus on precision here because of its especial importance to the model).

Apart from the model's inheriting from the more general class of quantum computers the property of being affected by precision issues[106], such issues have also specific relevance to the model by virtue of its *continua* (rather than discrete sets) of values. The author has, in conjunction with Rob Wagner at Leeds, undertaken preliminary work on the propagation of imprecision through continuous-variable operations such as 'displacement' and 'squeezing' (we comment that these operations represent low-level manipulations of coherent states, rather than relating in any direct way to high-level, desirable computations). The displacement operation, for example, suffers precision complexity of the order of $(\log p)^{-2}$, where, roughly speaking, p is a fixed, acceptable probability of wrongly measuring the displaced value (implementation—via the Micromaser system at Leeds—of values themselves is subject to a multiplicative error, due to technological limitations).

In summary, the resources relevant to ***continuous-variable quantum computers*** *are* ***time***, ***space*** *and* ***precision***.

Quantum Walks.

Aside. As well as specific models of quantum computation as discussed above, one may consider the computational use of *quantum walks*. There have been made suggestions (see, for example, [59, 60]) that there may be an increase in computational efficiency to be derived from quantum walks, though there have also been made suggestions ([85]) that *precision* costs are being overlooked in achieving this increase. The approach to complexity described in the present project would seem, then, to be a suitable formalization of these issues; we defer this to future work.

[106] Issues of precision affect the foundations of quantum theory itself, moreover. We have in mind here the controversy concerning the significance of precision's being necessarily finite in real-life computations—see Sect. 5.4.

3.6.3 Summary

We summarize in Table 3.1 the resources identified for each computational paradigm discussed in Sects. 3.5.1, 3.6.1 and 3.6.2.

Computational Model:	**Resources**:
Non-quantum computers (Sect. 3.6.1)	
Actual, physical implementations of Turing machines	Time, space, precision
Analogue/kinematic computers	Time, space, precision, energy
Relativistic computers	Time (as measured by the user), space, auxiliary computer's costs
Optical computers	Time, space, precision, energy
Chemical/DNA computers	Time, space, precision, energy, mass
Quantum computers (Sect. 3.6.2)	
Circuit-model	Time, space, precision
Adiabatic	Time, space, precision, energy
Measurement-based	Time, space, precision
Continuous-variable	Time, space, precision
Arbitrary computers (Sect. 3.5.1)	
Arbitrary	Time, space (also for *virtually* all models: precision, material cost, thermodynamic cost)

Table 3.1: A summary of the resources identified for consideration for each computational paradigm discussed in Sects. 3.5.1, 3.6.1 and 3.6.2.

We reiterate that, in light of Remark 31, we seek to identify *candidate* resources of interest for the computational models discussed above. Though this is not a rigorous process, neither does it need to be: in Chap. 4, we discuss notions that allow identification from among several candidate resources of those that are truly relevant to a computation; it is rather during this latter identification that rigour is desirable (and, indeed, present).

3.7 The Old versus the New

For the most part[107], the distinction between *traditional* resources—time and space[108]—and *unconventional* resources—such as those discussed above in this chapter—is seemingly fairly artificial: only in the extents to which they have been studied do the old/new resources seem to differ (since a Turing-machine-centric complexity theory warrants study of traditional, and only traditional, resources).

Contrastingly, we attempt to implement the present project's approach—including defining complexity functions, classes, etc. in a way homogeneous with respect to the resource under consideration—such that there is no distinction between the old and new resources. We retain the old as special cases, but use 'special' in the sense of 'more specific' rather than 'privileged'.

[107] For a notable potential exception, see Sect. 3.7.2.

[108] We suggest in Sect. 3.7.1 below that there exist no other conventional resources.

3.7.1 Two Conventional Resources

We argue now that, in order to capture the entirety of a conventional computer's complexity behaviour, one need consider only the resources of *time* and *space* (and variations thereon[109])—i.e., resources tailored specifically to and catering for Turing machines and similar.

Aside. We note in passing that, the present project's formalization of 'resource' notwithstanding, the above seems not to be a claim that succumbs to rigorous proof. Compare the situation with that of the Church-Turing thesis: once one formulates a workable, rigorous definition to capture the intuitive notion of 'effective computability', then one may prove equivalence with Turing-computability, adding weight to *but not offering proof of* the Church-Turing thesis; similarly, once one formulates a workable, rigorous definition of 'resource', then one has already built in the truth or otherwise of the above claim—the claim concerns an intuitive idea of resource, which may or may not be captured by the formal definition (for which definition, for what it is worth, one *may* be able to prove or disprove the claim).

The claim is corroborated by the practice of standard complexity theory, informed as it is by decades of experience of abstract theoreticians as well as of those considering complexity theory as it relates to real-world computation. We recall, for example, Păun's belief (see [104]) that

> "[t]he standard dimensions of computations are time and space"[110];

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, $\mathsf{P/poly}$, BPP, BQP and PP) or
- using a certain amount of *space* (loosely, PSPACE, AC^0, NC and L).[111]

Aside. 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

[109] Two notable variations on space and time respectively are *ink*—the number of writes to tape cells, regardless of whether the cells have been used previously—and *head reversals*—the number of occurrences of the tape head's moving to the right when its previous movement was to the left, or vice versa. Considering the amounts of these resources consumed during a halting Turing-machine computation, we have the following bounds: space $\leq$ ink $\leq$ time $\leq 2^{\text{space}}$; head reversals $\leq$ time. We defer further detail to the standard complexity literature, e.g., [37, 103].

[110] 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 in the present work do, question what resources are involved in unconventional computation.

[111] The 14 classes mentioned here are those in the 'Petting Zoo' section of Scott Aaronson's Complexity Zoo [1] (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 and SZK, which nonetheless are not defined in terms of resources other than time and space).

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. 3.2.2) must be mentioned in a list of conventional resources.

Time versus Space.

We summarize (and thereby do inadequate justice to) a long-standing contemplation of the fundamental difference between the resources of *time* and *space* by noting that space is reusable—tape cells may, once their contents are no longer needed, be overwritten—but time is not. We recall from [131] a way in which space can be contrived (with a view to shedding light on this fundamental difference) so as to better resemble time: suppose that each tape cell initially contains a zero that can during computation be changed to a one *but never back again* (we may imagine this as a punch-card system where the intact, zero state can be 'punched' to form a one-state hole, which cannot be refilled); then space, like time, is not reusable. Such devices are able to decide arithmetic predicates and nothing more—a strict subset of the standard Turing machine's repertoire.

3.7.2 Moore's Law

Though we comment above that, by and large, the distinction between conventional and unconventional resources arises only artificially (in that the former have been studied to a greater extent than have the latter), there is a potential notable exception: adherence to Moore's law.

Advances in the development of processors and memory suggest an exponential improvement (per unit cost) in conventional resources as implemented for use by digital computers. Famously, we have Moore's law:

> "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least ten years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65 000" [98].

This claim refers to transistors per integrated circuit (and specifically their number's doubling every two years), but has as a consequence that computing performance per unit cost—with respect to the resources of both time and space[112]—also doubles every two years.

Similarly, we recall from [75] Hendy's law, that the number of pixels per dollar offered by commercially available digital cameras—a rough indication of the increase in optical sensors' resolution and, hence, in the availability of the resource of precision—doubles every 18 to 19 months.

[112] In particular, the law relates directly to space as implemented as *random-access memory*; the analogous result concerning *hard-disc* space is provided by Kryder's law—see [130].

With the exception of precision, unconventional resources seem not to obey laws (akin to Moore's, Hendy's or Kryder's) of exponentially increasing availability. This may well be due to lack of financial investment: since digital computers process the vast majority of real-world computations, the efficient supply of conventional resources alone appears to be of economically (instead of academically) motivated interest. Rather than a significant, natural distinction between the conventional and unconventional, then, adherence or otherwise to such laws may be a mere manifestation of the artificial distinction discussed above.

3.8 Underlying Resource

In light of the existence of trade-offs between resources[113] (see Sects. 3.8.1 and 4.3.4), one may question whether there exists a single, all-encompassing, fundamental resource of which other specific resources are 'facets'. It is the view of Terry Rudolph [113] that this resource (should one exist) may be *low entropy*; further, this view seems to be in agreement with that of Karoline Wiesner—recall [53] and the first aside of Sect. 3.5.1 *Thermodynamic Cost*—insomuch as both views acknowledge the (computational) value of ordered structure.

Whilst we note that it is beyond the scope of the present project fully to discuss the existence or otherwise of a fundamental resource, we nonetheless suggest that the approach to complexity expounded here (and, in particular, the formalization of the notion of resource) may shed some light on the matter.

Figure 3.10: *Blind Monks Examining an Elephant* [73] by Hanabusa Itchō (1652 – 1724). Colour woodcut depicting several monks, each aware of—and basing his understanding of the whole elephant on—only a limited part of the elephant.

[113]Such trade-offs exist between resources not only during *computational processes* but also during *storage and retrieval of values*. Consideration of a value's Kolmogorov complexity (see Sect. 6.1.2) suggests an example of this latter form of trade-off: space consumption can be minimized by storing a value as its smallest generating program, at the cost of its retrieval from this program potentially taking a long time, and its storage—which entails discovery of the program—even longer (recall that such discovery is not in general a decidable problem. . .).

3.8.1 Trade-Offs

Space and Time.

In standard, Turing-machine complexity theory, the resources of time and space are related by the inequalities $s \leq t \leq 2^s$, where s is the number of tape cells written to, and t the number of time steps that elapse, during a (halting) Turing-machine computation (recall Footnote 109 for additional such inequalities). Beyond this, however, there seems to be little scope in general to trade off time for space or vice versa: whilst there may be specific problems admitting alternative Turing-machine solutions that, respectively, are time-efficient but space-intensive and space-efficient but time-intensive, it is not generally true that, for an arbitrary problem, one is able to engineer at will which of time and space is consumed in acceptably small quantity (neither would one expect this ability given the fundamental differences between time and space alluded to in Sect. 3.7.1 *Time versus Space*).

Precision and Space.

Between *unconventional* resources, on the other hand, there is more scope for trade-offs. Perhaps most obviously, there is for many computational models a trade-off between space and precision: by scaling up a computer's apparatus, more space but less precision is required; by scaling down, the onus is transferred in the opposite direction. See Sect. 4.3.4, and in particular Theorem 17, for further detail concerning this trade-off.

One may envisage other trade-offs between these two resources: consider an implementation of a Turing machine such that the ith tape cell ($i \in \mathbb{N}$) is allocated 2^{-i} units of physical storage space. Then the physical space occupied by the tape is at most two units, though the precision with which the tape must be written to/read from increases exponentially in the number of cells used.

Precision and Time.

Recall from Example 3 the Differential Analyzer; [43] alludes to two trade-offs between time and precision:

- > "Usually [a certain gear ratio] will be picked to be as small as possible, to cut down the solution time, noting, however, that there should be a substantial number of revolutions of every bus shaft of the machine in order to preserve precision."

- Further, a method is discussed whereby successive approximations give greater precision than is made directly available by the system: an initial, approximate plot of the desired function is made, from which can be identified an approximately equal and easily implementable function; the *difference* between this and the desired function is then plotted, and so on. Several such iterations incur a time cost in exchange for greater precision.

Further, we recall from Remark 33 José Félix Costa's observation that a balance scale tips increasingly slowly as the compared masses approach equality (i.e., as the precision required of the scale increases).

We note also (from [30]) that, in the context of real-number calculations performed by Turing machine and where exact solution (as opposed to approximation within some additive ϵ) is impossible, tractability may be modelled as run-time's being polynomial not in input size, but in $|\log \epsilon|$ *plus input size*. We have, then, a tacitly assumed trade-off between time and precision.

Energy and Time.

We remark in Sect. 3.6.2 *Adiabatic Quantum Computers* that the usual exposition (e.g., [64]) of adiabatic quantum computation presents time as the only relevant resource, though determining the time sufficient for a computation to succeed makes use of trade-offs with other resources such as energy.

As a particular example, we recall from [54] that there exists a quantum-adiabatic method for performing N-item database searches that can be contrived so as to run in constant time (cf. Grover's algorithm, which takes time in $\mathcal{O}\left(\sqrt{N}\right)$), but at the cost of requiring $\mathcal{O}\left(\sqrt{N}\right)$ energy, thus hinting, in this instance, at a $1:1$ trade-off between time and energy.

Relatedly, we recall that the example on page 205 of [94] relates the time taken for a quantum system to evolve into a highly correlated state and the energy spacings in an initial perturbation.

3.8.2 Combining Resources

A standard approach to complexity theory is to consider the class of problems that can be solved given a *single* resource bound: one may consider possible computations given, say, only polynomial time or logarithmic space, but rarely does one consider the *intersection* of these classes, where both time and space are bounded. In the present project's context of unconventional computation, however, there are under consideration many resources; whereas intersections of time- and space-based classes may not be of particular interest[114], it is prima facie feasible that study of intersections of classes based on other resources may be worthwhile[115]. However, we suggest now that this is not the case.

We conjecture in particular that, if there is apparent value in considering the combination of two resources, then this is only because there exists a *trade-off* between the resources being combined; in this case, there may be a more fundamental resource of which those combined are facets (i.e., functions), as discussed at the beginning of Sect. 3.8. Whereas considering several resources together may yield more information than considering them separately, this is only—according to the above conjecture—insomuch as this extra information is subsumed by knowledge of the trade-offs between the resources.[116]

[114] In fact, such intersections are treated, e.g., in [100, 121], though there is seemingly a dearth of such treatment elsewhere. This may be due to the facts that, in the Turing-machine case, there are interesting results concerning each resource individually and, hence, no need to lose information by combining the resources; and that, in the unconventional-computer case, there are many resources other than time and space and, hence, no reason to single out for combination the particular pair 'time and space'.

[115] We thank András Salamon for making this observation [115] and for bringing to our attention [121].

[116] Of course, even if this is true, then it does not mean that considering resources together is redundant—it may be a way of *deriving* information about trade-offs, for example.

We continue, then, to treat resources in isolation, other than when considering trade-offs, etc.; and we introduce in Chap. 4 a notion—*dominance*—that supersedes potential use of resource combination as a means of quantifying computers' overall complexity.

3.9 Summary

We comment above that analyses of unconventional computers often fail to capture the true complexity behaviour of the systems because the relevant resources are not considered. With the problem stated in this form, the solution becomes obvious: we need merely to consider the relevant resources. We see in this chapter, via our discussion of various interpretations of 'resource' and of various example resources[117] (that are relevant to various computational paradigms), how this may be done.

However, having solved one problem, we introduce another. With many resources under consideration, there are correspondingly many complexity functions, and, hence, it is no longer clear how to *compare* the complexity of computers. One may assess the overall complexity of a *Turing machine* by considering its time complexity (since the only relevant resources are time and space, and since time is always consumed in the greater quantity); and one may compare the overall complexity of two such machines using the pre-ordering '$\in \mathcal{O}$' of time-complexity functions. However, it is not clear how to quantify (whence to compare) the overall complexity of *unconventional computers*, with their many complexity functions. We address this difficulty in the following chapter.

[117] As a notable and detailed example, we discuss in Sect. 3.3 the resource of *precision*.

Chapter 4

Dominance

4.1 Comparison of Computers

In the realm of traditional, Turing-machine-style computation, the task of assessing computers' relative efficiency is well understood and correctly practised, specifically by comparing (using the $\lesssim$ pre-ordering—see Definition 21) *time* complexity functions: if $T^*_\Phi \lesssim T^*_\Psi$, then we may say that Φ is at least as efficient as Ψ—see Sect. 4.1.3.

In contrast with this, however, we have seen that the context of *unconventional* computation necessitates consideration of many different resources (both traditional—time, space, etc.—and non-traditional—precision, energy, etc.) in order to facilitate insightful complexity analyses. This renders more difficult the problem of *comparison* of computers' complexity (in which problem we seek to determine which of two given computers is the more efficient), simply because there are more candidate complexity functions potentially to compare; it is no longer a case, in particular, merely of $\lesssim$-comparing a certain, a priori chosen complexity function (e.g., that corresponding to the resource of time), since we no longer know which of the many complexity functions to $\lesssim$-compare. One 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.

We define and discuss below the notion of *dominance* 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 $\lesssim$-comparing computers' respective consumptions of *dominant/relevant* resources. (We introduce dominance in [25] and discuss the notion further in [21–24, 28]; much of this chapter is based upon these papers.)

4.1.1 The Need to Compare Computers

We motivate now the comparison of computers' respective efficiency.

Efficient Solution.

An obvious motivation for being able to compare the respective efficiency of two or more computing systems (that solve the same problem[1]) is that such ability allows one to ascertain which computer offers the more/most efficient solution to the problem. Especially if solution of the problem is to be implemented in real life, it is clearly desirable that implementation be at minimal (or at worst acceptably low) cost.

Complexity: Problems versus Solution Methods.

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 requires $\mathcal{O}(f)$ time, $\mathcal{O}(g)$ space, etc.[2]'.

However, it is typically 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 $\mathcal{O}(f)$ resource'

(a seemingly inherently elusive statement), it is usually forthcoming only that

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

(a relatively easily verifiable statement, at least once appropriate resources are considered).

Speaking more complexity-theoretically and problem-centrically than in the previous motivation (Sect. 4.1.1 *Efficient Solution*), then, one may wish to be able to ascertain which of several computers that solve the same problem[3] is the most efficient for the following reason.

> *The complexity of a* problem *is bounded above by the complexity of the best known* solution method *that solves the problem.*

The truth of this statement becomes clear upon consideration of the constituent terms' definitions: a *problem's* complexity is the minimal resource (as a function of input size) required to solve the problem, whereas a *solution method's* complexity is the (exact) resource required by the method to solve the problem; therefore, an arbitrary solution method for a problem witnesses that the problem's complexity is no greater than the arbitrary method's complexity, and the most informative such bound is given by the most efficient solution method.

In order to identify this method and, hence, this bound, it is therefore desired to be able to compare the efficiency of computers.

(Moreover, little more is typically known about a problem's complexity than that it is bounded above by that of the most efficient known solution method.)

[1] See Sect. 4.1.2.

[2] The 'etc.' here refers, of course, to computational resources other than run-time and memory space—see Chap. 3.

[3] See Sect. 4.1.2.

Note that increasing to a superset the set of considered solution methods can improve or leave unchanged (but never diminish) the efficiency of the most efficient member of the set; hence, the corresponding upper bound to the problem's complexity is, by consideration of a superset, either tightened (i.e., lessened) or left unchanged. The more comprehensive a set of considered solution methods for a problem, then, the more accurate the bound on the problem's complexity.[4] One desires, then, to consider *model-heterogeneous* sets of solution methods: if we are to attempt to include in our set as many methods as is possible, we wish not to restrict ourselves to methods conforming to a single computational paradigm (thereby imposing an unnecessary constraint on the size of sets of methods that can be considered, and a corresponding constraint on the quality of bounds on the complexity of problems). If the set of methods can contain computers from different models—*if we are able not only to compare computers, but further to compare computers of different types*—, then we may reasonably expect better bounds on the complexity of problems. Only when we can meaningfully compare, for example, the respective complexity functions of a Turing machine and a DNA computer can we begin to consider larger, model-heterogeneous sets of solution methods, and hence to obtain improved bounds on the complexity of problems; as we see below, our means of comparison is implemented such that such comparison *is* possible.

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—see Sect. 4.1.3; unfortunately, the same cannot be said for *non-standard* computers —see Sect. 4.1.4.

Turing-Machine Benchmarks.

Further motivation for the ability to compare computers—in particular computers from different computational models—comes from unconventional computing: practitioners of unconventional computation—those who either produce real-life instantiations of non-standard computers or investigate theoretically the properties of such devices—may very well wish to compare the efficiency of their non-standard devices with that of existing Turing machines (or digital-computer implementations thereof) that solve the same problem[5], since there is little worth (little *practical* worth, at least) in deviating from the well-studied, well-understood standard-computation realm in cases where the non-standard confers no computational/efficiency advantage.

One specific and eminently useful comparison that may be made, then, is between a novel unconventional computer and the benchmark of an existing, conventional computer that solves the same problem.

[4] Of course, a large set of inefficient methods offers a worse bound than a singleton set containing an efficient method, hence our discussing super-, rather than merely larger, sets.

[5] See Sect. 4.1.2.

4.1.2 Problem-Homogeneity

We comment that the computers compared in the way described (so as to determine which is the most efficient) will typically all solve the same problem; notably, the three motivations for comparison outlined in Sect. 4.1.1 all feature the phrase "that solve the same problem" (at Footnotes 1, 3 and 5).

Whilst it is *possible* to compare the respective efficiency of computers that solve different problems—specifically, our method of comparison neither knows nor cares which, or how many, are the problems that the computers solve—, the *relevance* of such comparison is unclear and its meaningfulness questionable.

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: such comparison is a poor way, therefore, 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* (see, e.g., [103]). Similarly, one may wonder what can be said about comparisons between computational *models*[6] rather than between individual computers, though the presently discussed method of comparison is not the means whereby such assessment can be made.

4.1.3 Comparing Turing Machines

We recap now the well-studied way in which the respective efficiency of standard computers may be compared. We begin by defining a pre-ordering of functions, which we employ, specifically, as a pre-ordering of *complexity* functions.

Definition 21.

- Write '$f \lesssim g$' for '$f \in \mathcal{O}(g)$';
- write '$f \not\lesssim g$' for '$f \notin \mathcal{O}(g)$'.

(Recall from Sect. 1.2.1 *Asymptotic Notation* the definition of $\mathcal{O}$-notation: $f \in \mathcal{O}(g)$ if and only if there exist a threshold n_0 and constant c such that, for all natural $n > n_0$, $|f(n)| \leq c\,|g(n)|$. Say that such a pair (n_0, c) *witnesses* that $f \in \mathcal{O}(g)$.)

Lemma 12. If functions f and g are non-negative and such that $f \leq g$ (i.e., if $0 \leq f(n) \leq g(n)$ for all $n \in \mathbb{N}$), then $f \lesssim g$.

Proof. If $f(n) \leq g(n)$ for all $n \in \mathbb{N}$, then certainly for all natural $n > 0$ $f(n) \leq g(n)$ (whence, by non-negativity of f and g, $|f(n)| \leq |g(n)|$). Hence, $(0, 1)$ witnesses that $f \lesssim g$. □

Lemma 13. $\lesssim$ is a pre-ordering of functions; i.e., $\lesssim$ is both reflexive and transitive.

[6] We comment as an aside that, whereas 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—, there exists in the literature a model-comparison approach, taken notably by Udi Boker and Nachum Dershowitz [33], for example.

Proof. For any f, $(0,1)$ witnesses that $f \lesssim f$, since $|f(n)| \leq |f(n)|$ for all natural $n > 0$; hence, $\lesssim$ is reflexive.

If (n_0, c) witnesses that $f \lesssim g$ and (n'_0, c') that $g \lesssim h$, then, for any natural $n > \max\{n_0, n'_0\}$, $|f(n)| \leq c\,|g(n)|$ and $|g(n)| \leq c'\,|h(n)|$, whence $|f(n)| \leq cc'\,|h(n)|$; that is, $(\max\{n_0, n'_0\}, cc')$ witnesses that $f \lesssim h$. Hence, $\lesssim$ is transitive. □

Of specific interest here is that $\lesssim$, a pre-ordering of functions, may be used as a pre-ordering of *complexity* functions (one may perform a 'like-with-like' comparison of the 'A-efficiency' of computers Φ and Ψ by observing which of $A^*_\Phi \lesssim A^*_\Psi$ and $A^*_\Psi \lesssim A^*_\Phi$ are true; comparisons of the form $A^*_\Phi \lesssim B^*_\Psi$ for *distinct* resources A and B are of more questionable significance, though we develop this idea below to advantage). We can now describe the method by which the efficiency of standard computers is compared.

Consider (conventional, Turing-machine-style) computers Φ and Ψ with respective time-complexity functions T^*_Φ and T^*_Ψ (see Definition 18). 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 (strictly) more efficient computer;
- if $T^*_\Phi \not\lesssim T^*_\Psi \lesssim T^*_\Phi$, then Ψ is deemed the (strictly) more efficient;
- 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.

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 Sect. 3.7.1), 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—recall Footnote 109 of Chap. 3).

4.1.4 Comparing Unconventional Computers

Whereas, in the Turing-machine case, time complexity encapsulates overall complexity, this is not necessarily true of unconventional computers consuming unconventional resources; it is certainly untrue, for example, of the factorization systems of Chap. 2, of which the polynomial time complexity indicates nothing of the systems' exponential overall complexity.

Note that, once overall complexity can be reliably measured and encapsulated in a single complexity function, then computers—whether standard or unconventional—can be compared by applying the $\lesssim$ pre-ordering to their respective overall-complexity functions; so as to solve the problem of computers' comparison, then, we focus below on the task of defining overall complexity. This is done by way of a criterion—*dominance*—that determines whether a resource is 'relevant' to a computer; an overall-complexity function may then be defined simply as the sum of those complexity functions that correspond to dominant resources.

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 their respective time-complexity functions. We describe in Sect. 4.2 a more suitable method, based on dominance, whereby such computers'[7] efficiency can be meaningfully compared.

Greatest Common Divisor.

We motivate more concretely, now, the notion of dominance. Supposing that we wish to find the *greatest common divisor* of two given, positive 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 complexity polynomial in n (the former by Lemma 11.7 of [103], the latter since $S^*_{\mathrm{E}}(n) \leq T^*_{\mathrm{E}}(n)$ for all n), and precision complexity constant (by Theorem 1, evocable since E is a traditional algorithm suitable for implementation by Turing machine) in n; and

- the *analogue system* (which we denote by 'B'[8]) of Sect. 3.3.2 *Greatest Common Divisor*, which has time and space complexity polynomial, and precision complexity exponential, in n (recall Sect. 3.3.2 *Greatest Common Divisor—Complexity*).

(The computational-complexity onus, it seems, can be transferred from one resource to another by selecting different solution methods.)

One may accurately describe the methods E and B as polynomial-time (and infer by 'like-with-like' comparison of time complexity that the methods are of comparable time-efficiency); however, it is intuitively more insightful to describe B as an *exponential-precision* (and, hence, less efficient overall), rather than *polynomial-time*, method, since the former description focuses on the more 'relevant' (i.e., more hungrily consumed) resource. B's comparable time efficiency notwithstanding, then, E seems more efficient overall: a *polynomial* consumption of 'relevant resource' is preferable to an *exponential* one.

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, of whether they conform to different computational paradigms, etc.

[7] Of course the computers so compared may instead be Turing machines, in which case their overall- and time-complexity functions coincide (since the only non-trivially consumed resources are time and space—recall Sect. 3.7.1—, of which only time is dominant—see Footnote 109 of Chap. 3). Further, the computers may be from the same computational model or from different ones—recall Sect. 4.1.1 *Complexity: Problems versus Solution Methods*.

[8] Much as our notation 'E' stands for 'Euclid', 'B' here stands for 'Blakey'; we hope that this vanity is mitigated by the fewness of the readers of this dissertation (and by the impracticability of the system). See also Footnote 11.

4.2 Dominance Defined

Here, we propose the notion of *dominance* as a way of augmenting $\mathcal{O}$-notation (and, specifically, the $\lesssim$ pre-ordering) so as to be able to compare computing systems' complexity. In particular, our augmentation allows meaningful comparison of the respective complexity functions, with respect to *different resources* (what matters is not that the resources be the same, but that they be *relevant*), of computing systems that conform to *different computational models*, thus allowing consideration of larger, model-heterogeneous sets of solution methods and, hence, improved bounds on problems' complexity (see Sect. 4.1.1 *Complexity: Problems versus Solution Methods*).

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 22 (provisional; see Definition 23)**.** 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 ("... for any resource B..."). It is not sufficient (in order that precision, for example, is shown to be dominant according to Definition 22) 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 functions, of which resources there are indeterminately many—this is clearly a futile task and a correspondingly worthless definition. Accordingly, we redefine dominance below *relative to an explicit set of resources.*

- Secondly, Definition 22 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 $f \not\lesssim g \not\lesssim f$.[9] 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 23 (to replace Definition 22)**.** Let Φ be a computer, and let $\mathcal{R}$ be a finite, non-empty set of resources, each consumed by Φ. 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 $B^*_\Phi \lesssim A^*_\Phi$.

Note that, whereas we stipulate that each element of $\mathcal{R}$ be 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'.

From this definition, then, we have that A is $\mathcal{R}$-dominant if and only if each $B \in \mathcal{R}$ is either

[9] However, it would be an immense surprise to the author were the old definition not to suffice in this respect for actual, practical computing systems—f and g satisfying $f \not\lesssim g \not\lesssim f$ seem necessarily to be contrived and unnatural.

- incomparable with A (in that $A^* \not\lesssim B^* \not\lesssim A^*$),
- strictly less than A (in that $A^* \not\lesssim B^* \lesssim A^*$), or
- equivalent to A (in that $A^* \lesssim B^* \lesssim A^*$);

that is, no $B \in \mathcal{R}$ is strictly greater than A (whence we should have that $A^* \lesssim B^* \not\lesssim A^*$).

By considering for a given computer only the *dominant* resources, we focus on the relevant measures for the computer—dominance formalizes a resource's relevance in that resources that are dominant impose the asymptotically greatest cost, to the extent that non-dominant resources may be disregarded as irrelevant. Further, we can *compare* computers according to their relevant (i.e., dominant) resources (using the '$\lesssim$' pre-ordering), as described below. We therefore have a framework in which can be made meaningful and consistent comparisons of computation-model-heterogeneous sets of computers (recall Sect. 4.1.1 *Complexity: Problems versus Solution Methods*); the framework can accommodate instances of various models of computation, and provide structure according to cost in terms of various resources.

Example 7. Recall that, in the complexity analysis of the factorization systems of Chap. 2, we suggest that precision, upon which the systems have an exponential dependency, is 'more relevant' than either time or space, upon each of which they have only a polynomial dependency. This can be formalized by noting that (for both systems) $T^*, S^* \lesssim P^*$, but neither $P^* \lesssim T^*$ nor $P^* \lesssim S^*$ (where, recall, P, T and S stand respectively for the resources of precision, time and space). Letting $\mathcal{R}$ denote the set $\{P, T, S\}$, then, precision—and precision alone—is $\mathcal{R}$-dominant for these systems.

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

Proposition 17. Let Φ and $\mathcal{R}$ be as in Definition 23, and let X and Y be elements of $\mathcal{R}$. 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 only the 'X $\mathcal{R}$-dominant $\Rightarrow$ Y $\mathcal{R}$-dominant' direction. The converse argument differs only in that the roles of X and Y are interchanged, which is valid since the hypotheses of the proposition are symmetrical in X and Y.) Suppose that X is $\mathcal{R}$-dominant; then, by Definition 23, $X^*_\Phi \lesssim B^*_\Phi \Rightarrow B^*_\Phi \lesssim X^*_\Phi$ 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. □

4.2.1 Overall Complexity

We define also $\mathcal{R}$-*complexity*, which, by summing those resources—and only those resources—that matter (in the eyes of Definition 23), offers in a single complexity function a measure of a computer's *overall complexity*.

Definition 24. Let Φ be a computer, and let $\mathcal{R}$ be a finite, non-empty set of resources for Φ. The *$\mathcal{R}$-complexity* of Φ (for size function[10] 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

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

Note that the superscript '$*$' in '$\mathcal{B}^*$' is included for uniformity with the general complexity-function notation 'A^*' (Definition 18), and should not be interpreted as suggesting that there exists a corresponding *resource* $\mathcal{B}$.[11]

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 25. 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 $\mathcal{B}^*_{\mathcal{R},\Phi,\sigma} \lesssim \mathcal{B}^*_{\mathcal{R},\Psi,\sigma} \not\gtrsim \mathcal{B}^*_{\mathcal{R},\Phi,\sigma}$;
- 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 $\mathcal{B}^*_{\mathcal{R},\Phi,\sigma} \lesssim \mathcal{B}^*_{\mathcal{R},\Psi,\sigma} \lesssim \mathcal{B}^*_{\mathcal{R},\Phi,\sigma}$; and
- the respective efficiency/complexity of Φ and Ψ is *incomparable*, denoted '$\Phi \wr_{\mathcal{R},\sigma} \Psi$' (or '$\Phi \wr \Psi$' if $\mathcal{R}$ and σ are understood), if $\mathcal{B}^*_{\mathcal{R},\Phi,\sigma} \not\lesssim \mathcal{B}^*_{\mathcal{R},\Psi,\sigma} \not\lesssim \mathcal{B}^*_{\mathcal{R},\Phi,\sigma}$.

We write '$\Phi \lesssim \Psi$' (or '$\Phi \lesssim_{\mathcal{R},\sigma} \Psi$') for 'either $\Phi \lesssim \Psi$ or $\Phi \simeq \Psi$'.

The relation '$\lesssim$' pre-orders computers: both reflexivity and transitivity are inherited from the corresponding relation between functions. More explicitly, $\Phi \lesssim \Phi$ (in fact, $\Phi \simeq \Phi$) for all Φ since $\mathcal{B}^*_\Phi \lesssim \mathcal{B}^*_\Phi$; and, if $\Phi \lesssim \Psi \lesssim \Xi$, then $\mathcal{B}^*_\Phi \lesssim \mathcal{B}^*_\Psi \lesssim \mathcal{B}^*_\Xi$, whence $\mathcal{B}^*_\Phi \lesssim \mathcal{B}^*_\Xi$, whence $\Phi \lesssim \Xi$.

Remark 36. One may question why we do not define our notion of 'overall complexity'—which we encapsulate in Definition 24 via summation of dominant resources—by summing *all* resources consumed by a computer. The chief reason is our not restricting very stringently what a resource actually is; whilst this allows consideration and accommodation of diverse models of computation and the diverse resources consumed thereby, it also necessitates some selectiveness with resources (see Sect. 4.5 for details of an additional restriction of the notion of resource) in order successfully to utilize the notion of dominance. This problem of having to manage potentially overwhelming resultant resource sets is alleviated by our defining overall (that is, $\mathcal{R}$-) complexity as the sum of $\mathcal{R}$-dominant, rather than arbitrary, resources.

Remark 37. As a final note in Sect. 4.2.1, we comment that it is in some contexts beneficial to retain the distinction between—and consider individually all complexity functions associated with—the various resources consumed by

[10] Recall from Sect. 3.4.3 *Size Functions* that we do not focus here on the choice of size function, instead taking such choice as read and building upon it.

[11] Whilst on the theme of notation, we make an apology similar to that of Footnote 8 for $\mathcal{B}$'s standing for 'Blakey'.

a computer, rather than to consider only its overall complexity. For example, though time (T) is for Turing machines always $\{T, S\}$-dominant (where S is space), there is still much to be gained in classical complexity theory by considering Turing machines' space complexity—hence the continued study of classes such as PSPACE, AC^0, NC and L, mentioned in Sect. 3.7.1.

4.3 Dominance Classes

4.3.1 Definitions

Given the above definitions of $\mathcal{R}$-dominance (Definition 23) and $\mathcal{R}$-complexity (Definition 24), it is possible—natural, even—to define corresponding complexity classes. We do this now, and investigate their interrelations.

Definition 26. Let $\mathcal{R}$ be a finite, non-empty set of resources (as in Definition 23).

- 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[12] 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 $\mathsf{C}_{\mathcal{R}}(f) = \bigcup_{A \in \mathcal{R}} \mathsf{C}_{\mathcal{R}}(f, A)$.
- Let $\mathsf{NC}_{\mathcal{R}}(f) = \bigcup_{A \in \mathcal{R}} \mathsf{NC}_{\mathcal{R}}(f, A)$.
- Let $\mathsf{B}_{\mathcal{R}}(f)$ denote the complexity class of problems of which each is solved by some 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.

We investigate in Sects. 4.3.2, 4.3.3 and 4.3.4 the relationships (inclusions, etc.) between the above-defined classes (namely, $\mathsf{C}_{\mathcal{R}}(f, A)$, $\mathsf{C}_{\mathcal{R}}(f)$, $\mathsf{B}_{\mathcal{R}}(f)$, and their non-deterministic counterparts). Further, we establish in Sect. 4.3.5 results detailing the correspondence between this and the traditional hierarchy. Exploring this correspondence (which exploration we largely defer to future work) allows the restatement of known results and open problems concerning traditional complexity theory's classes in terms of the new classes of Definition 26, and vice versa (with insight hopefully being offered by such restatement); see Sect. 4.3.6.

4.3.2 Expected Results

Non-Determinism.

There are certain results that one could reasonably expect of any 'sensibly constructed' theory of complexity. Notably, we have the following connection between determinism and non-determinism.

[12] Note, then, that we model non-determinism as a feature of computational paradigms, rather than as a (commodity) resource function—recall Sect. 3.2.1.

Theorem 4. Let $\mathcal{R}$ be a set of resources, containing as an element A. Let f be 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 deterministic computers are deemed to be a special case of non-deterministic computers. For example, each DPSM is, by Definition 9, an NPSM. □

Note that, by Lemma 11, this theorem would remain true even had we defined NPSMs to be *strictly non-deterministic*, rather than *not-necessarily-deterministic*, PSMs.

4.3.3 Other Internal Results

Those results that may reasonably be expected of a theory of complexity aside, there are many other inclusion theorems, etc. concerning the classes introduced in Definition 26; we state and prove some now. We term such results 'internal' (to the hierarchy of new classes) so as to distinguish them from results concerning classes in the traditional hierarchy.

Complexity Classes' Parameters.

We investigate the effects of altering complexity classes' defining parameters, by considering such questions as 'is there a relation of inclusion between $\mathsf{C}_{\mathcal{R}}(f)$ and $\mathsf{C}_{\mathcal{S}}(f)$ when $\mathcal{R} \subseteq \mathcal{S}$?'.

We consider the classes

1. $\mathsf{C}_{\mathcal{R}}(f, A)$,
2. $\mathsf{NC}_{\mathcal{R}}(f, A)$,
3. $\mathsf{C}_{\mathcal{R}}(f)$,
4. $\mathsf{NC}_{\mathcal{R}}(f)$,
5. $\mathsf{B}_{\mathcal{R}}(f)$, and
6. $\mathsf{NB}_{\mathcal{R}}(f)$

of Definition 26. For each, we investigate the effects (in terms of the inclusion relation $\subseteq$) of altering

A. the resource-set parameter $\mathcal{R}$
   i. to a superset $\mathcal{S} \supseteq \mathcal{R}$,
   ii. to a disjoint set $\mathcal{S}$ ($\mathcal{R} \cap \mathcal{S} = \emptyset$), and
   iii. to a set $\mathcal{S}$ such that $\mathcal{R} \nsubseteq \mathcal{S} \nsubseteq \mathcal{R}$ and $\mathcal{R} \cap \mathcal{S} \neq \emptyset$;

B. the bounding-function parameter f

i. to a function g such that $f \lesssim g$, and

ii. to a function g such that $f \leq g$;[13] and

C. the ($\mathcal{R}$-dominant) bounded-resource parameter A

i. to an $\mathcal{R}$-dominant resource B such that $A^*_\Phi \lesssim B^*_\Phi$ for all Φ, and

ii. to an $\mathcal{R}$-dominant resource B such that $A^*_\Phi \leq B^*_\Phi$ for all Φ.

(Of course, C is applicable only to classes 1 and 2, since classes 3 – 6 take no bounded-resource parameter.)

For convenience of reference, we label the theorems of Sect. 4.3.3 according to the list enumerations above. For example, the theorem that answers the question 'is there a relation of inclusion between $\mathsf{C}_\mathcal{R}(f)$ and $\mathsf{C}_\mathcal{S}(f)$ when $\mathcal{R} \subseteq \mathcal{S}$?' is labelled '3Ai' (further, we use set notation to refer to plural such questions: '$\{3,4\}$Ai' stands for '3Ai and 4Ai', '$3\{\mathrm{A},\mathrm{B}\}$i' for '3Ai and 3Bi', etc.). We summarize these theorems in Table 4.1 below.

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

- first (in Theorems 5 – 9) by contrasting a *subset* $\mathcal{R}$ against a *superset* $\mathcal{S}$,
- secondly (in Theorem 10) by contrasting *disjoint* sets $\mathcal{R}$ and $\mathcal{S}$, and
- thirdly (in Theorem 11) by contrasting *non-disjoint*, *non-enveloping* sets $\mathcal{R}$ and $\mathcal{S}$.

Theorem 5 (1Ai)**.** If $\mathcal{S}$ is a resource set, and if $A \in \mathcal{R} \subseteq \mathcal{S}$, then $\mathsf{C}_\mathcal{S}(f, A) \subseteq \mathsf{C}_\mathcal{R}(f, A)$.

Proof. Suppose that problem π is in class $\mathsf{C}_\mathcal{S}(f, A)$. By Definition 26, 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. □

(The converse inclusion does not necessarily hold, of course: it may be that $A \in \mathcal{R} \subseteq \mathcal{S}$, but that $\mathsf{C}_\mathcal{S}(f, A) \not\supseteq \mathsf{C}_\mathcal{R}(f, A)$. For example, a problem π that requires of a deterministic computer constant space (S) and quadratic time (T) complexity satisfies $\pi \in \mathsf{C}_{\{S\}}(n, S) \setminus \mathsf{C}_{\{S,T\}}(n, S)$, and $S \in \{S\} \subseteq \{S, T\}$. As a byproduct, this example gives the fundamentally reassuring result that some classes of the form $\mathsf{C}_\mathcal{R}(f, A)$ are indeed non-empty.)

Theorem 6 (2Ai)**.** If $\mathcal{S}$ is a resource set, and if $A \in \mathcal{R} \subseteq \mathcal{S}$, then $\mathsf{NC}_\mathcal{S}(f, A) \subseteq \mathsf{NC}_\mathcal{R}(f, A)$.

Proof. The proof is identical to that of Theorem 5 (1Ai), though with classes $\mathsf{C}_\mathcal{R}$ and $\mathsf{C}_\mathcal{S}$ replaced by $\mathsf{NC}_\mathcal{R}$ and $\mathsf{NC}_\mathcal{S}$ respectively, and with computer Φ non-deterministic. □

[13] Here and in subsequent theorems, '$\leq$' as a relation between functions is to be interpreted *pointwise*.

As a consequence of the previous two theorems, we have the following two.

Theorem 7 (3Ai)**.** If $\mathcal{S}$ is a resource set, and if $\mathcal{R} \subseteq \mathcal{S}$, then $\mathsf{C}_{\mathcal{S}}(f) \subseteq \mathsf{C}_{\mathcal{R}}(f)$.

Proof. If $\pi \in \mathsf{C}_{\mathcal{S}}(f) \overset{\text{Defn. 26}}{=} \bigcup_{R \in \mathcal{S}} \mathsf{C}_{\mathcal{S}}(f, R)$, then $\pi \in \mathsf{C}_{\mathcal{S}}(f, A)$ for some $A \in \mathcal{S}$. Hence, by Theorem 5 (1Ai), $\pi \in \mathsf{C}_{\mathcal{R}}(f, A)$, and so $\pi \in \bigcup_{R \in \mathcal{R}} \mathsf{C}_{\mathcal{R}}(f, R) \overset{\text{Defn. 26}}{=} \mathsf{C}_{\mathcal{R}}(f)$. □

Theorem 8 (4Ai)**.** If $\mathcal{S}$ is a resource set, and if $\mathcal{R} \subseteq \mathcal{S}$, then $\mathsf{NC}_{\mathcal{S}}(f) \subseteq \mathsf{NC}_{\mathcal{R}}(f)$.

Proof. The proof is identical to that of Theorem 7 (3Ai), though with classes $\mathsf{C}_{\mathcal{R}}$ and $\mathsf{C}_{\mathcal{S}}$ replaced by $\mathsf{NC}_{\mathcal{R}}$ and $\mathsf{NC}_{\mathcal{S}}$ respectively, and with Theorem 6 (2Ai) evoked in place of Theorem 5 (1Ai). □

Theorem 9 ($\{5, 6\}$Ai)**.** If $\mathcal{S}$ is a finite 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 does not necessarily hold either that $\mathsf{NB}_{\mathcal{S}}(f) \subseteq \mathsf{NB}_{\mathcal{R}}(f)$, or 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 $\bar{\mathcal{R}} = \left\{ A_1, A_2, A_3, \ldots, A_{\left|\bar{\mathcal{R}}\right|} \right\}$. 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 $\bar{\mathcal{S}}$, $\lesssim$-dominated by some element of $\bar{\mathcal{S}}$—which element we may take as B_i); hence, there exists a mapping $d \colon \bar{\mathcal{R}} \to \bar{\mathcal{S}} \colon A_i \mapsto B_i$ (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 resources in $\bar{\mathcal{S}} \setminus \{B\}$); in particular, $d(A_i)^* \leq \mathcal{B}^*_{\mathcal{S}}$ for all $A_i \in \bar{\mathcal{R}}$. Hence, $\sum_{i=1}^{\left|\bar{\mathcal{R}}\right|} d(A_i)^* \leq \left|\bar{\mathcal{R}}\right| \mathcal{B}^*_{\mathcal{S}}$, whence $\mathcal{O}\left(\sum_{i=1}^{\left|\bar{\mathcal{R}}\right|} d(A_i)^*\right) \subseteq \mathcal{O}\left(\left|\bar{\mathcal{R}}\right| \mathcal{B}^*_{\mathcal{S}}\right) = \mathcal{O}(\mathcal{B}^*_{\mathcal{S}})$; so $\mathcal{B}^*_{\mathcal{R}} = \sum_{i=1}^{\left|\bar{\mathcal{R}}\right|} A_i^* \overset{\text{Lemma 1}}{\in} \mathcal{O}\left(\sum_{i=1}^{\left|\bar{\mathcal{R}}\right|} d(A_i)^*\right) \subseteq \mathcal{O}(\mathcal{B}^*_{\mathcal{S}})$. $\mathcal{B}^*_{\mathcal{R}} \in \mathcal{O}(\mathcal{B}^*_{\mathcal{S}})$, 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 $\mathcal{B}^*_{\mathcal{R}} \colon n \mapsto 10$ and $\mathcal{B}^*_{\mathcal{S}} \colon n \mapsto n^2$. 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 $f \colon n \mapsto 20$ and $f' \colon n \mapsto 2n^2$. 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. □

We consider now the alteration of resource set $\mathcal{R}$ to a disjoint set $\mathcal{S}$.

Theorem 10 ($\{1, 2, 3, 4, 5, 6\}$Aii)**.** 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.

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[14] of $\mathcal{R}$ and $\mathcal{S}$ 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, which latter is not necessarily space here).[15] 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 (including positive-degree polynomials) 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. □

We consider now the alteration of resource set $\mathcal{R}$ to a set $\mathcal{S}$ such that $\mathcal{R} \nsubseteq \mathcal{S} \nsubseteq \mathcal{R}$ and $\mathcal{R} \cap \mathcal{S} \neq \emptyset$. As we see below, this reduces to the previous case in which $\mathcal{R}$ and $\mathcal{S}$ are disjoint.

Theorem 11 ($\{1, 2, 3, 4, 5, 6\}$Aiii)**.** 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

[14] Clearly $\mathcal{R}$ and $\mathcal{S}$ are disjoint if and only if $\mathcal{S}$ and $\mathcal{R}$ are.

[15] Such computers exist: letting R be precision and S time, any exponential-time Turing machine satisfies these criteria by Theorem 1.

- $\mathsf{NB}_{\mathcal{R}}(f)$ 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 of Definition 17. Since the classes ($\mathsf{C}_{\mathcal{R}}(f, A)$, $\mathsf{NC}_{\mathcal{R}}(f, A)$, $\mathsf{C}_{\mathcal{R}}(f)$, $\mathsf{NC}_{\mathcal{R}}(f)$, $\mathsf{B}_{\mathcal{R}}(f)$ and $\mathsf{NB}_{\mathcal{R}}(f)$) and function ($\mathcal{B}^*_{\mathcal{R}}$) under consideration 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 constructed here, 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 $\mathcal{R} \cap \mathcal{S} \neq \emptyset$. That no inclusion or ordering holds in generality for $\mathcal{R}'$ and $\mathcal{S}'$ (by the previous theorem) gives that none holds for $\mathcal{R}$ and $\mathcal{S}$ as constructed here. □

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

- first (in Theorems 12 – 14) by contrasting functions *related by* $\lesssim$, and
- secondly (in Theorem 15) by contrasting functions *related by* $\leq$.

Theorem 12 ($\{1,2\}\mathrm{B}\{\mathrm{i},\mathrm{ii}\}$)**.** 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, $f \lesssim g$ (recall Lemma 12), and so the conclusions still hold.

Proof. ($1\mathrm{B}\{\mathrm{i},\mathrm{ii}\}$.) 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 $\mathsf{C}_{\mathcal{R}}(g, A)$.

($2\mathrm{B}\{\mathrm{i},\mathrm{ii}\}$.) Similarly, if a problem is in $\mathsf{NC}_{\mathcal{R}}(f, A)$ by virtue of its being solved by non-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 $\mathsf{NC}_{\mathcal{R}}(g, A)$, as required. □

Consequently, we have the following.

Theorem 13 ($\{3,4\}\mathrm{B}\{\mathrm{i},\mathrm{ii}\}$)**.** 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 26, $\mathsf{C}_{\mathcal{R}}(f) = \bigcup_{R \in \mathcal{R}} \mathsf{C}_{\mathcal{R}}(f, R)$. By $|\mathcal{R}|$ evocations of Theorem 12 (1Bi), this is a subset of $\bigcup_{R \in \mathcal{R}} \mathsf{C}_{\mathcal{R}}(g, R)$, which, by Definition 26, is $\mathsf{C}_{\mathcal{R}}(g)$, as required.

That $\mathsf{NC}_{\mathcal{R}}(f) \subseteq \mathsf{NC}_{\mathcal{R}}(g)$ follows in an analogous way from evocations of Theorem 12 (2Bi). □

Theorem 14 ($\{5,6\}$Bi)**.** 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, then, by any computer, optimal for the problem that it solves, with an identity time-complexity function $T^*(n) = n$, which necessarily exists[16] by the time-constructibility of polynomials. □

However, the following restriction yields a more definite result than Theorem 14 ($\{5,6\}$Bi).

Theorem 15 ($\{5,6\}$Bii)**.** 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 $\mathsf{B}_{\mathcal{R}}(f)$ [respectively $\mathsf{NB}_{\mathcal{R}}(f)$] is solved by a deterministic [respectively non-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 $\mathsf{B}_{\mathcal{R}}(g)$ [respectively $\mathsf{NB}_{\mathcal{R}}(g)$]. □

Resource Parameter. Finally, we consider (in Theorem 16) 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 16 ($\{1,2\}$C$\{$i, ii$\}$)**.** 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 $\mathsf{C}_{\mathcal{R}}(f,B)$ [respectively $\mathsf{NC}_{\mathcal{R}}(f,B)$] is solved by a deterministic [respectively non-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 $\mathsf{C}_{\mathcal{R}}(f,A)$ [respectively $\mathsf{NC}_{\mathcal{R}}(f,A)$] by virtue of Φ. □

For a brief summary of the above theorems, see Table 4.1.

[16] Take for example a Turing machine that returns the parity of the length of its input string—fewer than n time steps are insufficient to distinguish the parities of the lengths of strings '$x_1x_2x_3\ldots x_n$' and '$x_1x_2x_3\ldots x_n1$', and so a Turing machine with $T^*(n) = n$ is optimal for this task.

	Ai: $\mathcal{R} \subseteq \mathcal{S}$	**Aii**: $\mathcal{R} \cap \mathcal{S} = \emptyset$ **Aiii**: $\mathcal{R} \nsubseteq \mathcal{S} \nsubseteq \mathcal{R}$, $\mathcal{R} \cap \mathcal{S} \neq \emptyset$	**Bi**: $f \lesssim g$	**Bii**: $f \leq g$	**Ci**: $A^* \lesssim B^*$ **Cii**: $A^* \leq B^*$
1: $\mathsf{C}_{\mathcal{R}}(f, A)$	$\mathsf{C}_{\mathcal{S}} \subseteq \mathsf{C}_{\mathcal{R}}$ 5	$\mathsf{C}_{\mathcal{R}} \pitchfork \mathsf{C}_{\mathcal{S}}$ 10/11	$\mathsf{C}(f, A) \subseteq \mathsf{C}(g, A)$ 12	$\mathsf{C}(f, A) \subseteq \mathsf{C}(g, A)$ 12	$\mathsf{C}(f, B) \subseteq \mathsf{C}(f, A)$ 16
2: $\mathsf{NC}_{\mathcal{R}}(f, A)$	$\mathsf{NC}_{\mathcal{S}} \subseteq \mathsf{NC}_{\mathcal{R}}$ 6	$\mathsf{NC}_{\mathcal{R}} \pitchfork \mathsf{NC}_{\mathcal{S}}$ 10/11	$\mathsf{NC}(f, A) \subseteq \mathsf{NC}(g, A)$ 12	$\mathsf{NC}(f, A) \subseteq \mathsf{NC}(g, A)$ 12	$\mathsf{NC}(f, B) \subseteq \mathsf{NC}(f, A)$ 16
3: $\mathsf{C}_{\mathcal{R}}(f)$	$\mathsf{C}_{\mathcal{S}} \subseteq \mathsf{C}_{\mathcal{R}}$ 7	$\mathsf{C}_{\mathcal{R}} \pitchfork \mathsf{C}_{\mathcal{S}}$ 10/11	$\mathsf{C}(f) \subseteq \mathsf{C}(g)$ 13	$\mathsf{C}(f) \subseteq \mathsf{C}(g)$ 13	(N/A)
4: $\mathsf{NC}_{\mathcal{R}}(f)$	$\mathsf{NC}_{\mathcal{S}} \subseteq \mathsf{NC}_{\mathcal{R}}$ 8	$\mathsf{NC}_{\mathcal{R}} \pitchfork \mathsf{NC}_{\mathcal{S}}$ 10/11	$\mathsf{NC}(f) \subseteq \mathsf{NC}(g)$ 13	$\mathsf{NC}(f) \subseteq \mathsf{NC}(g)$ 13	
5: $\mathsf{B}_{\mathcal{R}}(f)$	$\mathcal{B}^*_{\mathcal{R}} \lesssim \mathcal{B}^*_{\mathcal{S}}$, $\mathsf{B}_{\mathcal{R}} \pitchfork \mathsf{B}_{\mathcal{S}}$ 9	$\mathcal{B}^*_{\mathcal{R}} \pitchfork \mathcal{B}^*_{\mathcal{S}}$, $\mathsf{B}_{\mathcal{R}} \pitchfork \mathsf{B}_{\mathcal{S}}$ 10/11	$\mathsf{B}(f) \pitchfork \mathsf{B}(g)$ 14	$\mathsf{B}(f) \subseteq \mathsf{B}(g)$ 15	
6: $\mathsf{NB}_{\mathcal{R}}(f)$	$\mathcal{B}^*_{\mathcal{R}} \lesssim \mathcal{B}^*_{\mathcal{S}}$, $\mathsf{NB}_{\mathcal{R}} \pitchfork \mathsf{NB}_{\mathcal{S}}$ 9	$\mathcal{B}^*_{\mathcal{R}} \pitchfork \mathcal{B}^*_{\mathcal{S}}$, $\mathsf{NB}_{\mathcal{R}} \pitchfork \mathsf{NB}_{\mathcal{S}}$ 10/11	$\mathsf{NB}(f) \pitchfork \mathsf{NB}(g)$ 14	$\mathsf{NB}(f) \subseteq \mathsf{NB}(g)$ 15	

Table 4.1: A brief summary of the class-inclusion theorems of Sect. 4.3.3. 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 at the bottom-right of cells indicate the corresponding theorems.

4.3.4 Trade-Off Results

One intuitively feels that there should be scope to trade off resources against each other (and, hence, that dominant resource can to a certain extent be engineered by careful selection of the computer used to solve a particular problem). An obvious such trade-off is between the resources of *space* and *precision*: simply by scaling down [respectively, up] a system's apparatus, required space is reduced but required precision increased [or vice versa] (recall Sect. 3.8.1 *Precision and Space*); this is now formalized.

Theorem 17. For any DPSM, there exists an equivalent[17] DPSM with constant space complexity (though possibly with precision complexity greater than that of the former DPSM). For any DPSM, there exists an equivalent DPSM with constant precision complexity (though possibly with space complexity greater than that of the former DPSM).

Similarly, for any NPSM, there exist an equivalent NPSM with constant space complexity and an equivalent NPSM with constant precision complexity.

Though the resource of precision has been formally defined (in Sect. 3.3.3) only for PSMs, similar definitions may be made for other computational models, whence theorems analogous to Theorem 17 may be proven (at least for those paradigms satisfying suitable assumptions regarding the continuity of space—see Remark 38).

Proof. Let Φ be a PSM that computes problem π. Let S denote the resource of space and P precision. Suppose that $S^*_{\Phi,\sigma} \lesssim f$ for some function f. Let Ψ be the PSM derived by scaling each of the three spatial axes of Φ by a factor of $\alpha := (f \circ \sigma(x))^{-\frac{1}{3}}$, for input value x; Ψ computes the same problem π as Φ since the miniaturization process by which the former system is derived does not alter the computation relation. The space $S_\Psi(x)$ required by Ψ on input x, then, is $\alpha^3 S_\Phi(x) = \frac{S_\Phi(x)}{f \circ \sigma(x)}$, and, consequently, the space complexity $S^*_{\Psi,\sigma}(n)$ of Ψ evaluated at input size n is $\frac{S^*_{\Phi,\sigma}(n)}{f(n)}$; since $S^*_{\Phi,\sigma} \lesssim f$, $S^*_{\Psi,\sigma}$ is bounded above by a constant.[18]

In scaling the apparatus of Φ to produce Ψ, we have similarly scaled the input and output parameters of Φ. Each such parameter either relies on spatial precision (in which case it contributes a scaling of required precision by a factor of $\frac{1}{\alpha}$) or it does not (in which case it contributes no change to required precision).[19] Letting k be the number of input/output parameters of Φ reliant on spatial precision, $P^*_{\Psi,\sigma}(n) = f(n)^k P^*_{\Phi,\sigma}(n)$; in reducing the size of the system's apparatus, we have made constant the space complexity but increased by a factor of f^k the precision complexity.

[17] Equivalence here is of the computers' respective computation relations $\smile_\Phi$.

[18] The exponent $-\frac{1}{3}$ in the definition of α, and our cubing α in the expression for $S_\Psi(x)$, reflect our assumption that Φ and Ψ inhabit a *three*-(spatial-)dimensional universe; if, more generally, our computational model renders these systems in n dimensions, then the exponent becomes $-\frac{1}{n}$, and cubing is replaced by raising to the power of n.

[19] In fact, each parameter relying *additively* on precision contributes a scaling of $\frac{1}{\alpha}$, whereas each relying *multiplicatively* contributes no change (the reader is invited to confirm this); we describe, then, the *worst-case* increase in precision complexity (assuming that all errors are either additive or multiplicative—investigation of more exotic errors' contributions is beyond the present project's scope).

This argument, though with the roles of S and P reversed, shows that, by expanding apparatus, precision complexity can be made constant at the expense of increased space complexity.

Note that the system Φ *before* making this trade-off (in either direction) is deterministic if and only if the system Ψ *after* making it is deterministic—scaling the apparatus does not alter the number of possible responses of the system to any given configuration. Hence, letting Φ be deterministic, we obtain the first part of the theorem's claim; letting it be non-deterministic, we obtain the second. □

Remark 38. Note that assumptions such as those concerning the divisibility of space (essentially that space is continuous, allowing arbitrary miniaturization $\Phi \mapsto \Psi$ of computers) are accounted for by considering precision complexity (and, of less interest here, precision-related manufacturing costs). The measure of complexity tells us how precise one needs to be for the computer to function as expected, and technology, the enveloping physics, etc. determine whether this precision is feasible.

Corollary 3. Let f, g and k be such that there exists a deterministic [respectively, non-deterministic] computer with space complexity f, precision complexity g and k input/output parameters reliant (in an additive way—see Footnote 19) on spatial precision. Write P for precision, S for space and $\mathcal{R}$ for $\{P, S\}$. Then

$$\mathsf{C}_{\mathcal{R}}(f, S) \cup \mathsf{C}_{\mathcal{R}}(g, P) \subseteq \mathsf{C}_{\mathcal{R}}\left(f^k g, P\right)$$

$$[\text{respectively, } \mathsf{NC}_{\mathcal{R}}(f, S) \cup \mathsf{NC}_{\mathcal{R}}(g, P) \subseteq \mathsf{NC}_{\mathcal{R}}\left(f^k g, P\right)].$$

(Of particular interest here are the inclusions $\mathsf{C}_{\mathcal{R}}(f, S) \subseteq \mathsf{C}_{\mathcal{R}}\left(f^k g, P\right)$ and $\mathsf{NC}_{\mathcal{R}}(f, S) \subseteq \mathsf{NC}_{\mathcal{R}}\left(f^k g, P\right)$.)

Proof. Letting Φ, Ψ and π be as in the proof of Theorem 17, and writing g for P^*_Φ, we have that Φ is as the deterministic [respectively, non-deterministic] computer in the hypotheses of this corollary; as in the proof of Theorem 17, we have precision and space complexity as summarized in the following table.

	Φ	Ψ
S^*	$\mathcal{O}(f)$	$\mathcal{O}(1)$
P^*	g	$f^k g$

Since one (at the fewest) of P and S is $\mathcal{R}$-dominant for Φ, we have that π is, by virtue of Φ, a member of $\mathsf{C}_{\mathcal{R}}(f, S) \cup \mathsf{C}_{\mathcal{R}}(g, P)$ [respectively, $\mathsf{NC}_{\mathcal{R}}(f, S) \cup \mathsf{NC}_{\mathcal{R}}(g, P)$]; since, from the above table, P is $\mathcal{R}$-dominant for Ψ, we have that π is, by virtue of Ψ, a member of $\mathsf{C}_{\mathcal{R}}\left(f^k g, P\right)$ [respectively, $\mathsf{NC}_{\mathcal{R}}\left(f^k g, P\right)$].

The structure of the proof of Theorem 17 (in particular, the quantification structure '$\forall\Phi\exists\Psi$') gives the class inclusions in the direction stated in the claim of this corollary. □

Aside. Recall from Sect. 3.8 that resource trade-offs exist by virtue not only of *computational processes* but also of values' *storage and retrieval*. We mention a specific example, concerning Kolmogorov complexity, in Sect. 6.1.2.

4.3.5 Relationship to Traditional Hierarchy

Theorem 18.

- $\mathsf{DTIME}(f) \subseteq \mathsf{C}_{\{T\}}(f, T)$.
- $\mathsf{NTIME}(f) \subseteq \mathsf{NC}_{\{T\}}(f, T)$.
- $\mathsf{DSPACE}(f) \subseteq \mathsf{C}_{\{S\}}(f, S)$.
- $\mathsf{NSPACE}(f) \subseteq \mathsf{NC}_{\{S\}}(f, S)$.

Proof. These class inclusions follow directly from Definition 26. We make this more explicit for the illustrative example of the first bullet point.

Suppose that a problem π is in $\mathsf{DTIME}(f)$; then it is so by virtue of a (deterministic) Turing machine M with time complexity $T^*_M \lesssim f$. Also by virtue of M, then, and since T is $\{T\}$-dominant for all computers (and in particular for M), $\pi \in \mathsf{C}_{\{T\}}(f, T)$ by Definition 26. □

Note that the converse inclusions would imply the unlikely situation in which Turing machines can simulate any computer, conforming to any paradigm, without an increase in time complexity and (separately) without an increase in space complexity.

Stronger (by Theorems 5 and 6 ($\{1, 2\}$Ai)) than the first and second bullet points of Theorem 18 is the following.

Theorem 19.

- $\mathsf{DTIME}(f) \subseteq \mathsf{C}_{\{T,S\}}(f, T)$.
- $\mathsf{NTIME}(f) \subseteq \mathsf{NC}_{\{T,S\}}(f, T)$.

Proof. Given the observation that T is $\{T, S\}$-dominant for all Turing machines, the proof is as that of Theorem 18. □

4.3.6 Transfer between Hierarchies of Results/Questions

We defer to future work a full investigation of the correspondence between the traditional and new hierarchies of complexity classes, but conclude this section by noting that, with such correspondence established, it becomes possible to restate

- *theorems* already proven in either hierarchy in terms of the classes of the other, thereby giving new, analogous theorems in the latter context; and
- *open problems* concerning either hierarchy in terms of the classes of the other, thereby potentially offering new approaches to these questions.

4.4 An Analogue of the Gap Theorem

As we comment above, one may attempt to establish the correspondence between traditional and new complexity classes, and thereby to translate known results from the old context to the new. Another approach to such translation, however, is *directly* to prove results concerning the new context, without necessarily having established the correspondence between the hierarchies.

We exemplify this latter approach now by proving, in the context of the complexity classes of Definition 26, an analogue of the *Gap Theorem*. We thank András Salamon for originally posing, in [114], the question that inspired [20], upon which much of this section is based.

4.4.1 Introduction

Thanks to Trachtenbrot [125] and Borodin [34], complexity theorists have the *Gap Theorem*, which guarantees the existence of arbitrarily large gaps in the (traditional) complexity hierarchy. That is, there can be made arbitrarily large increases in resource availability that, for all their vastness, yield no additional computational power, in that every computation that can be performed given the extra resource could have been performed in its absence.

As we see above, needs arising during the complexity analysis of unconventional computers can be addressed via the notion of *dominance*, with which comes a corresponding hierarchy of complexity classes—recall Definition 26—, and it is natural to ask whether a gap theorem, analogous to that of Trachtenbrot and Borodin, holds in this context.

We show below that a gap theorem does indeed hold for certain dominance-based classes. We consider also related statements concerning other dominance-based classes, and formulate corresponding conjectures.

Traditional Gap Theorem.

We introduce now the Gap Theorem, the existing result to which our Theorem 21 is analogous.

Computational tasks can be organized into a hierarchy according to the amount of some resource (A, say) required to perform the computations. The Gap Theorem, proven by Trachtenbrot [125] and, independently, Borodin [34], guarantees arbitrarily large gaps in this hierarchy; that is, the theorem guarantees arbitrarily large increases in the availability of resource A that nonetheless offer no new computable functions. (Note that this says nothing about consumption of resources *other than* A; we return to this point in Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—Resource Compensation.*)

The theorem is independent of choice of computational model (though does assume at most a countable infinity of computers; see Sect. 4.4.4 *Comments—Countability of Computers*), and relies only upon A's satisfying certain (natural and lenient) conditions, namely Blum's axioms (see Sect. 3.4.1 *Blum's Axioms*).

Theorem 20 (Gap Theorem)**.** If A is a resource that satisfies Blum's axioms, and if $g\colon \mathbb{N} \to \mathbb{N}$ is a computable function such that, for all $x \in \mathbb{N}$, $g(x) \geq x$, then there exists a computable function $t\colon \mathbb{N} \to \mathbb{N}$ such that every computer Φ satisfying $A^*_\Phi \leq g \circ t$ also satisfies $A^*_\Phi \leq t$.

Remark 39. The non-decreasing function g represents an increase (from t to $g \circ t$) in the availability of resource A.[20] The theorem guarantees that, for some t, such increase offers no additional computational power: if a computation can be performed using no more than $g \circ t(n)$ units of A when processing any input

[20]That this is indeed an increase follows from the fact that $g(x) \geq x$ for all $x \in \mathbb{N}$, whence $g \circ t$ represents a greater supply of—and a more lenient bound on—A than does t.

of size n, then it can in fact be performed using no more than $t(n)$ units. This fruitless increase is our eponymous gap.

Remark 40. As can be seen, the theorem is slightly stronger than is suggested by its preceding, informal statement ("every computation that can be performed given the extra resource could have been performed in its absence"): for computer Φ with resource bound $g \circ t$, not only is there *some* equivalent computer with bound t, but Φ *itself* has this property: $A^*_\Phi \leq t$.

We give a proof based on that offered by Borodin [34]; a similar approach is used in Sect. 4.4.2 *Proof of the Theorem* in the proof of our analogous statement concerning dominance classes (see Definition 26).

Proof. We assume some enumeration Φ_0, Φ_1, Φ_2, ... of computers (this is discussed further in Sect. 4.4.4 *Comments—Countability of Computers*).

For brevity, we write '$\mathrm{P}_n(i,j)$' for the property '$A^*_{\Phi_i}(n) \leq j$' $(i,j \in \mathbb{N})$ throughout this proof.

The proof is constructive; we define $t\colon \mathbb{N} \to \mathbb{N}$ inductively, as follows.

$$\begin{aligned} t(0) &:= 1\ ; \\ t(n) &:= \min\left\{ k \in \mathbb{N} \;\middle|\; \begin{array}{ll} & k > t(n-1) \\ \wedge & (\forall i \leq n)\left[\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)\right] \end{array} \right\} \end{aligned}$$

(for $n > 0$).

Note that the condition

$$\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k) \tag{4.1}$$

(that is, $A^*_{\Phi_i}(n) \leq g(k) \Rightarrow A^*_{\Phi_i}(n) \leq k$) is logically equivalent to $A^*_{\Phi_i}(n) > g(k) \vee A^*_{\Phi_i}(n) \leq k$, which hopefully gives some insight into why this choice of t might satisfy the theorem.

We show now, inductively, that this t is well defined. As the base case, it is clear that $t(0) := 1$ is well defined. As an inductive hypothesis, suppose that $t(n-1)$ is well defined, and consider $t(n)$. For any $i \leq n$, either

(a) the computation of Φ_i is defined at all inputs of size n, whence $A^*_{\Phi_i}(n) \in \mathbb{N}$, and so there exists (minimal) $k \geq A^*_{\Phi_i}(n)$ (whence the consequent of (4.1)—and, hence, (4.1) as a whole—is rendered true); or

(b) the computation of Φ_i is undefined at some input of size n, whence the condition $A^*_{\Phi_i}(n) \leq g(k)$ fails for all $k \in \mathbb{N}$, $A^*_{\Phi_i}(n)$ being undefined (whence the antecedent of (4.1) fails, rendering (4.1) as a whole true).

The maximum of all k as in case (a)—of which there are finitely many, whence this maximum's existence—(and of $t(n-1)+1$, which is, by the inductive hypothesis, well defined), then, is a member of

$$\left\{ k \in \mathbb{N} \;\middle|\; \begin{array}{ll} & k > t(n-1) \\ \wedge & (\forall i \leq n)\left[\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)\right] \end{array} \right\}\ ; \tag{4.2}$$

this set is a *non-empty* subset of $\mathbb{N}$, and, hence, has a well-defined minimum, as which we define $t(n)$.

t is well defined.

We claim now that t is computable. This follows since g is computable and since A satisfies the second of Blum's axioms, whence, for each n, (4.1) is a decidable predicate.

So,

$$t \text{ is well defined and computable};$$

what of the main property described in the theorem (i.e., that every Φ such that $A^*_\Phi \leq g \circ t$ in fact satisfies $A^*_\Phi \leq t$)?

Let Φ (which we suppose to be Φ_j in our enumeration of computers) be such that $A^*_\Phi \leq g \circ t$; that is,

$$\mathrm{P}_n\left(j, g \circ t\left(n\right)\right) \tag{4.3}$$

for all n. Fix $n \geq j$. By construction, $t\left(n\right)$ is a member (in fact, the minimum) of the set (4.2), and, hence, satisfies the set's defining property: $t\left(n\right)$ is such that $\left(\forall i \leq n\right)\left[\mathrm{P}_n\left(i, g \circ t\left(n\right)\right) \Rightarrow \mathrm{P}_n\left(i, t\left(n\right)\right)\right]$; in particular, since $j \leq n$, we have that $\mathrm{P}_n\left(j, g \circ t\left(n\right)\right) \Rightarrow \mathrm{P}_n\left(j, t\left(n\right)\right)$, which, recalling (4.3) and applying modus ponens, gives that $\mathrm{P}_n\left(j, t\left(n\right)\right)$, i.e., that $A^*_{\Phi_j}\left(n\right) \leq t\left(n\right)$. We have that our computer $\Phi = \Phi_j$, which we suppose only to have consumption of A bounded by $g \circ t$, in fact has consumption of A bounded by t, as required. □

(We reiterate that this proof is based on that given in [34].)

Note for comparison that, implicitly, the Gap Theorem (Theorem 20) concerns complexity classes of the form

$$\mathsf{S}\left(f\right) := \left\{ \text{ problem solved by } \Phi \mid A^*_\Phi \leq f \right\} \ ; \tag{4.4}$$

specifically, the theorem says that, for each suitable g, there exists some t such that $\mathsf{S}\left(g \circ t\right) = \mathsf{S}\left(t\right)$.

Of the classes of Definition 26, we suggest that $\mathsf{B}_\mathcal{R}\left(f\right)$ has more in common (than does either $\mathsf{C}_\mathcal{R}\left(f, A\right)$ or $\mathsf{C}_\mathcal{R}\left(f\right)$) with these Gap-Theorem classes $\mathsf{S}\left(f\right)$. Notably, $\mathsf{S}\left(f\right)$ and $\mathsf{B}_\mathcal{R}\left(f\right)$ both bound their members' complexity with a standard *inequality*, '$\leq f$', whereas $\mathsf{C}_\mathcal{R}\left(f, A\right)$ and $\mathsf{C}_\mathcal{R}\left(f\right)$ bound theirs with an *asymptotic-notation constraint*, '$\lesssim f$'. Because of this similarity (and others) between S and B, one may expect the proof method of Theorem 20 to be most easily adapted to demonstrate the validity of the analogous gap theorem for B (more easily, that is, than to demonstrate the validity of the respective gap theorems for $\mathsf{C}_\mathcal{R}\left(f, A\right)$ and $\mathsf{C}_\mathcal{R}\left(f\right)$); indeed, this appears to be the case: we prove Sect. 4.4's main result—the gap theorem for class $\mathsf{B}_\mathcal{R}\left(f\right)$—in this way.

4.4.2 Dominance Gap Theorem

Given the Gap Theorem (Theorem 20) of Trachtenbrot [125] and Borodin [34], and given the complexity classes defined in terms of dominance (Definition 26), it is natural to ask whether an analogous theorem holds for these classes.

We are led to consider several statements, one for each of $\mathsf{B}_\mathcal{R}\left(f\right)$, $\mathsf{C}_\mathcal{R}\left(f, A\right)$ and $\mathsf{C}_\mathcal{R}\left(f\right)$; Sect. 4.4's main result, which we give in Sect. 4.4.2 *Statement of the Theorem* and prove in Sect. 4.4.2 *Proof of the Theorem*, is the statement concerning $\mathsf{B}_\mathcal{R}\left(f\right)$ (the other two statements, along with potential approaches to their proof/refutation, are discussed in Sect. 4.4.3 *Related Statements*).

Statement of the Theorem.

Theorem 21 (Dominance Gap Theorem)**.** If $\mathcal{R}$ is a set of resources that satisfy Blum's axioms, and if $g\colon \mathbb{N} \to \mathbb{N}$ is a computable function such that, for all $x \in \mathbb{N}$, $g(x) \geq x$, then there exists a computable function $t\colon \mathbb{N} \to \mathbb{N}$ such that $\mathsf{B}_{\mathcal{R}}(g \circ t) = \mathsf{B}_{\mathcal{R}}(t)$.

Relationship to the Traditional Gap Theorem.

Before proving Theorem 21, we make some comments relating it to Theorem 20, etc.

Differences. Theorem 20 considers a single Blum resource, predicting arbitrarily large gaps in the hierarchy corresponding to this resource. Theorem 21 makes the same prediction, though about the hierarchy corresponding (via dominance) to a *set* of Blum resources.

'$|\mathcal{R}| = 1$' Case. If $\mathcal{R}$ is a singleton set (say $\mathcal{R} = \{A\}$, where A may or may not satisfy Blum's axioms), then A is vacuously $\mathcal{R}$-dominant for all computers; hence, $\mathcal{B}^*_{\mathcal{R}} = A^*$ and the statement of Theorem 21 becomes that of Theorem 20. In this sense, Theorem 21 generalizes Theorem 20.

If, in addition, A satisfies Blum's axioms, then Theorem 20, and, hence, Theorem 21, hold: there exist arbitrarily large gaps in the corresponding hierarchies.

(We consider this special case in order to illustrate the relationship between the traditional and new gap theorems, not as a step towards the general proof of the latter, for which see Sect. 4.4.2 *Proof of the Theorem.*)

Summation. If we were to consider the complexity hierarchy corresponding to a fixed, finite sum of (Blum) resources A_1, A_2, A_3, ..., A_l, then the analogous gap theorem holds:

> for any computable, non-decreasing g, there exists a computable t such that a problem's computability with $\left(\sum_{i=1}^{l} A_i\right)^* \leq g \circ t$ implies its computability with $\left(\sum_{i=1}^{l} A_i\right)^* \leq t$.

This follows from the fact that, if each A_i satisfies Blum's axioms, then $\sum_i A_i$ can be viewed as a (single) resource *that itself satisfies Blum's axioms*, whence we may evoke Theorem 20.

A notable instance of this is the situation of Theorem 21, where we consider the sum $\mathcal{B}^*_{\mathcal{R}} := \sum_{A \text{ is } \mathcal{R}\text{-dominant}} A^*$; we may apply Theorem 20 to this sum and obtain the desired result. However, note that the addends of this sum (specifically, their presence in or absence from the sum) are dependent on the computer considered: for $\Phi \neq \Psi$, it may well be the case that

$$\{\, A \mid A \text{ is } \mathcal{R}\text{-dominant for } \Phi \,\} \neq \{\, A \mid A \text{ is } \mathcal{R}\text{-dominant for } \Psi \,\} \quad ;$$

for this reason, it is more instructive and advantageous, when considering statements of the (slightly more general) form 'for each computer within resource bound $g \circ t$, there exists an *equivalent* computer within bound t', to pursue the proof method of Sect. 4.4.2 *Proof of the Theorem.*

Resource Compensation. Note that the (traditional) Gap Theorem says nothing about resources other than the one (A, say) considered; it is reasonable to entertain the possibility that the arbitrarily large increase in A yields no additional computing power *because of a corresponding decrease in other resources' availability.*

However, Theorem 21 seems to say more, in this respect, than Theorem 20: we sum *all* dominant resources; a dearth of some resources, seemingly, cannot be compensated for in the same way. Prima facie, then, Theorem 20 may seem more plausible than Theorem 21 (since the latter seems to say more than the former); nonetheless, both hold, and indeed their proofs are similar.

Generalization. We note above, in Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—'$|\mathcal{R}| = 1$' Case*, that Theorem 21 generalizes Theorem 20; there is a similar implication in Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—Resource Compensation.* We note in passing that the *proof method* of Theorem 20 allows further generalization: $t(n)$ can be defined to be

$$\min \{ k \in \mathbb{N} \mid k > t(n-1) \wedge (\forall i \leq n) [\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)] \}$$

for an *arbitrary* property P_n of pairs of natural numbers, leading, where this t is well defined, to results analogous to Theorem 20 (in which instance $\mathrm{P}_n(i,j)$ is $A^*_{\Phi_i}(n) \leq j$). We adopt this method, taking $\mathrm{P}_n(i,j)$ to be $\mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \leq j$, to prove Theorem 21.

Proof of the Theorem.

We recall from Sect. 4.4.2 *Statement of the Theorem* the statement of the Dominance Gap Theorem:

> If $\mathcal{R}$ is a set of resources that satisfy Blum's axioms, and if $g\colon \mathbb{N} \to \mathbb{N}$ is a computable function such that, for all $x \in \mathbb{N}$, $g(x) \geq x$, then there exists a computable function $t\colon \mathbb{N} \to \mathbb{N}$ such that $\mathsf{B}_{\mathcal{R}}(g \circ t) = \mathsf{B}_{\mathcal{R}}(t)$.

Proof. We closely follow the proof of Theorem 20. Once again, enumerate computers as Φ_0, Φ_1, Φ_2, ... Let $\mathcal{R}$ and g be as in the statement of the theorem. Let '$\mathrm{P}_n(i,j)$' denote the property '$\mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \leq j$' ($i, j \in \mathbb{N}$).

Define $t\colon \mathbb{N} \to \mathbb{N}$, as follows.

$$\begin{array}{rcl} t(0) & := & 1\ ; \\ t(n) & := & \min \left\{ k \in \mathbb{N} \;\middle|\; \begin{array}{ll} & k > t(n-1) \\ \wedge & (\forall i \leq n) [\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)] \end{array} \right\} \end{array}$$

(for $n > 0$).

(We have that the condition $\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)$ (that is, $\mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \leq g(k) \Rightarrow \mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \leq k$) is equivalent to $\mathcal{B}^*_{\mathcal{R},\Phi_i}(n) > g(k) \vee \mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \leq k$.)

We show by induction that t is well defined. As the base case, it is clear that $t(0) := 1$ is well defined. Suppose as an inductive hypothesis that $t(n-1)$ is well defined, and consider $t(n)$. For any $i \leq n$, either

(a) the computation of Φ_i is defined for all inputs of size n, whence $\mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \in \mathbb{N}$, and so there exists (minimal) $k \geq \mathcal{B}^*_{\mathcal{R},\Phi_i}(n)$; or

(b) the computation of Φ_i is undefined for some input of size n, whence the condition $\mathcal{B}^*_{\mathcal{R},\Phi_i}(n) \leq g(k)$ fails for all $k \in \mathbb{N}$ (since $\mathcal{B}^*_{\mathcal{R},\Phi_i}(n)$ is undefined).

The maximum of all k as in case (a) (and of $t(n-1)+1$, which is, by the inductive hypothesis, well defined), then, is a member of

$$\left\{ k \in \mathbb{N} \;\middle|\; \begin{array}{ll} & k > t(n-1) \\ \wedge & (\forall i \leq n)\left[\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)\right] \end{array} \right\};$$

this subset of $\mathbb{N}$ is non-empty, and, hence, has a well-defined minimum—$t(n)$—, as required.

t is well defined.

Further, t is computable. This is because g is computable and the members of $\mathcal{R}$ satisfy the second of Blum's axioms, whence '$\mathrm{P}_n(i, g(k)) \Rightarrow \mathrm{P}_n(i,k)$' is a decidable predicate.

We have that

t is well defined and computable;

we turn now to the main property described in the theorem: that every Φ such that $\mathcal{B}^*_{\mathcal{R},\Phi} \leq g \circ t$ satisfies $\mathcal{B}^*_{\mathcal{R},\Phi} \leq t$ (whence $\mathsf{B}_{\mathcal{R}}(g \circ t) = \mathsf{B}_{\mathcal{R}}(t)$, as required).

Let Φ (Φ_j, say, in our enumeration) be such that $\mathcal{B}^*_{\mathcal{R},\Phi} \leq g \circ t$, so

$$\mathrm{P}_n(j, g \circ t(n)) \tag{4.5}$$

for all n. Fix $n \geq j$.

By definition, $t(n)$ satisfies $(\forall i \leq n)\left[\mathrm{P}_n(i, g \circ t(n)) \Rightarrow \mathrm{P}_n(i, t(n))\right]$; in particular, since $j \leq n$, we have that $\mathrm{P}_n(j, g \circ t(n)) \Rightarrow \mathrm{P}_n(j, t(n))$, which, from (4.5) by modus ponens, gives that $\mathrm{P}_n(j, t(n))$, i.e., $\mathcal{B}^*_{\mathcal{R},\Phi_j}(n) \leq t(n)$. We have that $\mathcal{B}^*_{\mathcal{R},\Phi}(n) \leq t(n)$: Φ, which we suppose only to have $\mathcal{R}$-complexity bounded by $g \circ t$, in fact has $\mathcal{R}$-complexity bounded by t, as required. □

As described in Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—Generalization*, we have adapted the proof of Theorem 20 to show that our Dominance Gap Theorem for class $\mathsf{B}_{\mathcal{R}}$ holds.

4.4.3 Future Work

Related Statements.

Just as we have considered (and demonstrated the truth of) the gap theorem corresponding to the complexity class $\mathsf{B}_{\mathcal{R}}(f)$, we may consider those corresponding to $\mathsf{C}_{\mathcal{R}}(f, A)$ and $\mathsf{C}_{\mathcal{R}}(f)$. Specifically, we may ask which (if either) of the following statements hold.

Statement 1 (Dominance Gap Theorem, $\mathsf{C}_{\mathcal{R}}(f, A)$ variation)**.** If $\mathcal{R}$ is a set of resources that satisfy Blum's axioms, if $A \in \mathcal{R}$, and if $g\colon \mathbb{N} \to \mathbb{N}$ is a computable function such that, for all $x \in \mathbb{N}$, $g(x) \geq x$, then there exists a computable function $t\colon \mathbb{N} \to \mathbb{N}$ such that $\mathsf{C}_{\mathcal{R}}(g \circ t, A) = \mathsf{C}_{\mathcal{R}}(t, A)$.

Statement 2 (Dominance Gap Theorem, $\mathsf{C}_{\mathcal{R}}(f)$ variation)**.** If $\mathcal{R}$ is a set of resources that satisfy Blum's axioms, and if $g\colon \mathbb{N} \to \mathbb{N}$ is a computable function such that, for all $x \in \mathbb{N}$, $g(x) \geq x$, then there exists a computable function $t\colon \mathbb{N} \to \mathbb{N}$ such that $\mathsf{C}_{\mathcal{R}}(g \circ t) = \mathsf{C}_{\mathcal{R}}(t)$.

Though we defer the formal proof/refutation of these statements to future work, we nonetheless formulate and discuss here some preliminary conjectures towards such proof/refutation.

Conjecture 1. We suspect that neither Statement 1 nor Statement 2 holds in generality.

In attempting to prove Statement 1 or 2 *starting from Theorem 21 and its proof method*, there are two notable potential obstacles precluding the direct transfer from the $\mathsf{B}_{\mathcal{R}}$ result to the $\mathsf{C}_{\mathcal{R}}$ cases. These are

1. that $\mathsf{C}_{\mathcal{R}}(f, A)$ and $\mathsf{C}_{\mathcal{R}}(f)$ are defined in terms of the '$\lesssim$' pre-ordering, whereas $\mathsf{B}_{\mathcal{R}}(f)$ is defined in terms of '$\leq$'; and
2. that $\mathsf{C}_{\mathcal{R}}(f, A)$ and $\mathsf{C}_{\mathcal{R}}(f)$ are defined in terms of *existence* of some $\mathcal{R}$-dominant resource, whereas $\mathsf{B}_{\mathcal{R}}(f)$ is defined in terms of the *sum* of $\mathcal{R}$-dominant resources.

Note that obstacle 1 may be surmountable: it may be possible to introduce constants and thresholds (as in Definition 1), so as to convert between statements concerning $\mathsf{C}_{\mathcal{R}}$ and those concerning $\mathsf{B}_{\mathcal{R}}$-like classes. This especially seems feasible since it is not necessary that one constant/threshold pair work for all computers under consideration; when constructing t, we may instead find a pair for each computer and take a suitable maximum. (The implicit suggestion, then, is that t be defined so as to guarantee a gap sufficiently large to subsume the leeway conferred by $\mathcal{O}$-notation.)

Obstacle 2, however, seems harder to overcome. It is difficult, in particular, to demonstrate that an arbitrary Φ that solves a problem in $\mathsf{C}_{\mathcal{R}}(f)$ satisfies $\mathcal{B}^*_{\mathcal{R},\Phi} \leq g \circ f$ for non-trivial g, since $\mathcal{R}$ may contain $\lesssim$-incomparable, $\mathcal{R}$-dominant resources, rendering bounds on $\mathcal{B}^*_{\mathcal{R}}$ much more restrictive than those on single $\mathcal{R}$-dominant resources. A proof of Statement 1 or 2, then, would almost certainly bear little resemblance to those of Theorems 20 and 21.

We suspect that Statement 2, in particular, fails because of the following. Unlike $\mathsf{S}(f)$—recall (4.4)—, and assuming that $|\mathcal{R}| > 1$, $\mathsf{C}_{\mathcal{R}}(f)$ concerns more than one resource, and so we cannot directly use Theorem 20 to prove Statement 2. Fixing some computable, non-decreasing $g\colon \mathbb{N} \to \mathbb{N}$, Theorem 20 gives that there exists t such that $A^*_{\Phi} \leq g \circ t$ implies that $A^*_{\Phi} \leq t$. Supposing that Φ solves a problem in $\mathsf{C}_{\mathcal{R}}(g \circ t)$, we have that there exists an $A \in \mathcal{R}$ that is $\mathcal{R}$-dominant for Φ and that satisfies $A^*_{\Phi} \lesssim g \circ t$. Assuming that the solution proposed above to obstacle 1 allows us to state '$A^*_{\Psi} \leq g \circ t$' or similar for some Ψ equivalent to Φ (possibly subject to introduction and suitable handling of constants/thresholds), then, Theorem 20 gives that $A^*_{\Psi} \leq t$, whence $A^*_{\Psi} \lesssim t$. However, *A, whilst $\mathcal{R}$-dominant for Φ, may not be $\mathcal{R}$-dominant for Ψ*; we cannot conclude that Ψ (nor, hence, Φ) solves a problem in $\mathsf{C}_{\mathcal{R}}(t)$ (cf. Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—Summation*).

Conjecture 2. If Statement 1 holds for all A in some $\mathcal{R}$ (even if not in generality), then Statement 2 holds for this $\mathcal{R}$.

Conjecture 2 certainly does not follow trivially from the relevant definitions (notably that of $\mathsf{C}_{\mathcal{R}}(f)$), since Statement 1's holding for all $A \in \mathcal{R}$ guarantees only that, for each A, there exists (an *A-dependent*) t as described, and not necessarily that there exists a *single* t that behaves as described for every A (whence Statement 2 would hold for $\mathcal{R}$). However, one approach to a potential proof of this conjecture is to define $t(n)$ to be the minimum of a set (as in the proof of Theorems 20 and 21), though where the set is bounded below not only by $t(n-1)$ but also by suitable values corresponding to each $A \in \mathcal{R}$; by construction, then, we ensure that t is large enough simultaneously to work for all A.

As a nod towards a converse of Conjecture 2, if Statement 2 holds for $\mathcal{R}$, then one would expect Statement 1 to fail for some $A \in \mathcal{R}$ only when $\mathcal{R}$ is very contrived and unnatural (if at all). We defer formalization of this sentiment, along with further treatment of the above conjectures, to future work.

Conjecture 3. Statements 1 and 2 hold for all *singleton* sets $\mathcal{R}$.

Recall that, if $\mathcal{R}$ is a singleton set (say $\mathcal{R} = \{A\}$), then A is vacuously $\mathcal{R}$-dominant for any computer. In the case of $\mathsf{B}_{\mathcal{R}}$, this means that Theorems 20 and 21 coincide (see Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—'$|\mathcal{R}| = 1$' Case*). The implication is not as direct in the cases of $\mathsf{C}_{\mathcal{R}}(f, A)$ and $\mathsf{C}_{\mathcal{R}}(f)$, because of obstacle 1 above (though only this obstacle: obstacle 2 is, of course, not an issue when $|\mathcal{R}| = 1$); nonetheless, the constant/threshold suggestion following Conjecture 1 may again be applied, hence Conjecture 3.

Proposition 18. If Statement 1 fails for some computable, non-decreasing function g, then it fails for any computable, non-decreasing function g' such that $g' \geq g$.

Similarly, if Statement 2 fails for some computable, non-decreasing function g, then it fails for any computable, non-decreasing function g' such that $g' \geq g$.

Proof. (We prove the Statement-1 case; the proof of the Statement-2 case differs only in the omission of the ten occurrences of '$, A$' and in the evocation of Theorem 13 instead of Theorem 12.)

$g' \geq g$ gives that $\mathsf{C}_{\mathcal{R}}(g' \circ f, A) \supseteq \mathsf{C}_{\mathcal{R}}(g \circ f, A)$ (recall Theorem 12), and so $\mathsf{C}_{\mathcal{R}}(g' \circ f, A) \setminus \mathsf{C}_{\mathcal{R}}(f, A) \supseteq \mathsf{C}_{\mathcal{R}}(g \circ f, A) \setminus \mathsf{C}_{\mathcal{R}}(f, A)$. Hence, if $\mathsf{C}_{\mathcal{R}}(g \circ f, A) \setminus \mathsf{C}_{\mathcal{R}}(f, A) \neq \emptyset$, then $\mathsf{C}_{\mathcal{R}}(g' \circ f, A) \setminus \mathsf{C}_{\mathcal{R}}(f, A) \neq \emptyset$. □

Consequently, if looking for a counterexample to show that Statement 1 or 2 fails, then we may profitably consider *large* g; and, if trying to show by contradiction that Statement 1 or 2 holds, then we may profitably consider a *minimal* g for which the statement fails.

Other Future Work.

Aside from Statements 1 and 2 and related conjectures, a further question for future investigation is suggested when one recalls (e.g., from [82]) that the traditional Gap Theorem remains true when generalized to computations on sets of *ordinals*: does this generalized theorem hold in the context of dominance classes?

Yet another direction for future work is to ascertain conditions—on choice of resource and, more relevantly to the present context, on details of the way in

which dominance is implemented—that lead to hierarchy theorems (analogous to the time/space hierarchy theorems, which guarantee for every time/space-bounded complexity class a strictly larger time/space-bounded class) and to a suitable notion of 'resource-constructibility'.

4.4.4 Conclusion

Summary.

In Sect. 4.4.1, we recall the Gap Theorem of Trachtenbrot [125] and Borodin [34], which guarantees arbitrarily large gaps in the hierarchy of standard complexity classes $\mathsf{S}(f)$. We recall from Definition 26 dominance complexity classes $\mathsf{C}_{\mathcal{R}}(f, A)$, $\mathsf{C}_{\mathcal{R}}(f)$ and $\mathsf{B}_{\mathcal{R}}(f)$, with a view to investigating whether analogous gap theorems hold for these classes.

In Sect. 4.4.2, we demonstrate that the Dominance Gap Theorem (that is, the gap theorem corresponding to the hierarchy of classes of the form $\mathsf{B}_{\mathcal{R}}(f)$) holds.

In Sect. 4.4.3, we consider (amongst other directions for future work) the gap theorems for $\mathsf{C}_{\mathcal{R}}(f, A)$ and $\mathsf{C}_{\mathcal{R}}(f)$, conjecturing that, in generality, they fail.

Comments.

Countability of Computers. Recall that the proofs of Theorems 20 and 21 rely on there being an enumeration Φ_i ($i \in \mathbb{N}$) of computers, and, hence, on their countability. This is clearly the case with Turing machines: each such machine can, due essentially to the finite nature of its description, be uniquely encoded as a natural number. In a sense, however, this seems not to be the case for certain other computers. Consider, for example, a family $\{\, \Phi_x \mid x \in \mathbb{R} \,\}$ (or even just $\{\, \Phi_x \mid x \in [0, 1] \,\}$) of analogue computers where Φ_x adds x to its (real-number) input by mechanical means; there is clearly, in theory at least, a continuum of such computers. However, the storage and retrieval of real numbers to infinite precision is an abstraction of the analogue computational model that cannot be realized in practice; whatever our achievable precision when dealing with real numbers, we are left with a sub-family of 'realizable' Φ_x, which *are* enumerable.

General Comments. From a *practical* viewpoint, the chief result of Sect. 4.4 is Theorem 21; from a *theoretical* viewpoint, one significant result concerns the scope of the theorem's proof, on which we now comment.

The proof method of the Gap Theorem has a broader applicability than is commonly acknowledged; this we have seen abstractly in Sect. 4.4.2 *Relationship to the Traditional Gap Theorem—Generalization*, and concretely in the proof of Theorem 21. This unrealized potential is understandable, since the method's typical use—namely, proving the Gap Theorem—suffices for standard, Turing-machine-type complexity theory. When analyzing the complexity of *non-standard* computers, however, and, in particular, when considering such computers' $\mathcal{R}$*-complexity*, we may nonetheless adapt the proof method in order to demonstrate the validity of the analogous gap theorem.

What we note is that the spirit of the Gap Theorem holds in contexts other than that in which it was originally formulated. The statement of the theorem

is limited *artificially* so as to reflect common usage of the tools of complexity theory, not *of necessity* by the proof's scope. In particular, this observation yields Theorem 21.

More generally and fundamentally, however, note that the Gap Theorem offers an example that supports an underlying thesis, namely that:

when analyzing the complexity of unconventional *computers, we should use* unconventional *complexity-theoretic techniques (or at the very least check that the standard, Turing-machine-type techniques still work in the same way).*

What we note from Theorem 21 is that the existing tool of the Gap Theorem *can* still be evoked in the context of dominance (specifically, in the context of the hierarchy of classes $\mathsf{B}_{\mathcal{R}}$).

Aside. We comment above that our consideration of the Gap Theorem exemplifies the practice of verifying that traditional complexity-theoretic techniques work also in an unconventional-computing setting. The theorem is a suitable choice for such verification due to its fame within complexity theory and its continued relevance to researchers (e.g., [114]) more than 40 years after its proof; on the other hand, we could (and, arguably, would) have chosen to verify a complexity-theoretic tool that is routinely and directly *used* more than the Gap Theorem. Whereas the Gap Theorem is an important part of the edifice of complexity theory and has other results built thereupon, a technique such as *reduction*, for example, is used directly—rather than merely being built upon—on a daily basis by complexity theorists.

In any case, the notions of unconventional-computer complexity theory expounded, and related approaches advocated, in the present work allow verification of whichever tools and techniques are required by the practitioners of non-standard computation who chose to adopt aspects of the present work.

4.5 Normalization—Resources Revisited

We see above that dominance offers a means of defining a computer's overall complexity (Definition 24), and thence a means of comparing computers' efficiency (Definition 25). We show below, however, that, for the notion of dominance to function in an expected and desirable way, we need to be more stringently restrictive than above—notably, more restrictive than are Blum's axioms—in our notion of resource (Sect. 3.4).

The content of this section owes much to [23].

4.5.1 The Problem: Dominance versus 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.[21] As far as Sect. 3.4 (which, in particular, stipulates that

[21] Note that our consideration of the resources of *space* and *time* is intended to be illustrative of general, non-resource-specific concepts as explored in the present work. The choice of these particular resources is arbitrary, and we do not, in particular, rely here or in the related discussion in Sect. 4.5.2 on results—$S \leq T \leq 2^S$, etc.—that depend upon properties of the specific nature of these resources.

Blum's axioms be satisfied) is concerned, S is 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 $\iota\colon S(x) \mapsto S'(x)$ 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 (via ι) resources with which to quantify space usage.

Now, if we make inter-resource comparisons according to dominance (recall Sect. 4.2), then we may find that S' dominates T (more precisely, that S' is $\{S', T\}$-dominant and T is not), but that T dominates S (i.e., 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 linear space and quadratic time complexity.

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. That is,

one 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.

We stress that each of S and S' is, in isolation, unproblematic; it is these measures' relative significance that behaves questionably.

(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 defined in terms of dominance) suffer a corresponding, knock-on unreliability.)

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

Remark 41. Note that the problem described here—that the impact of the 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 *understated* by considering, say, $\lceil \log S \rceil$ instead of S. Our answer to the former problem (namely, normalization—see Sect. 4.5.2) 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, may 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.

4.5.2 The Solution: Normalization

To resolve inconsistencies such as that above, in which space can, depending upon how it is measured (or even upon how measured values are labelled), seem either more or less important than time, we introduce a notion of *normality* of resource, advocating consideration only of *normal* resources during non-standard

computers' complexity analyses. We do this in such a way that S and T above are deemed normal (and hence their mutual comparison is deemed valid), but that 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[22] to S; our definition satisfies this.

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. The choice of this particular property as our notion of normality is motivated as follows. Note that S and S' in the above example 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 $\{2^0, 2^1, 2^2, \ldots\}$; the former property seems the more natural (quite literally!), and we use it as the model of our notion of normalization.

We now formalize these ideas.

Definition of Normalization.

Definition 27. Let $\mathcal{C}$ be a class of computers.[23] For each $\Phi \in \mathcal{C}$, let X_Φ be the set of input values for Φ.

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

$$A^{\mathcal{C}}_\Phi(x) := |\{\, A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_\Phi(x) \,\}| \quad .^{24} \tag{4.6}$$

- Resource 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 $x \in X_\Phi$).

Example of Normalization.

We give now an example, which relates to previous discussion, of this definition's use. Recall from our example (in Sect. 4.5.1) the resources S, S' and T, 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.

Proposition 19. $S'^{\mathcal{T}} = S^{\mathcal{T}} = S$.

Proof. 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,

$$\{\, S_\Psi(y) \mid \Psi \in \mathcal{T} \wedge y \in X_\Psi \,\} = \mathbb{N} \ , \tag{4.7}$$

[22] *Normalization* is a process whereby non-normal resources can be converted into normal resources that are order-isomorphic with the originals.

[23] $\mathcal{C}$ may, for example, be the class of Turing machines, or a class of analogue computers (such as are introduced in [107, 117]). This is not to say that $\mathcal{C}$ necessarily corresponds to a computational model, though it may well.

[24] 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 $A_\Phi(x)$.

and so, by definition of S',

$$\{ S'_\Psi(y) \mid \Psi \in \mathcal{T} \wedge y \in X_\Psi \} = \{ 2^n \mid n \in \mathbb{N} \} \quad . \tag{4.8}$$

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

$$\begin{array}{rcl} S'^{\mathcal{T}}_\Phi(x) & \overset{(4.6)}{=} & |\{ S'_\Psi(y) \mid \Psi \in \mathcal{T} \wedge y \in X_\Psi \wedge S'_\Psi(y) < S'_\Phi(x) \}| \\ & \overset{(4.8)}{=} & |\{ 2^n \mid n \in \mathbb{N} \wedge 2^n < S'_\Phi(x) \}| \\ & \overset{\text{defn. of } S'}{=} & \left|\left\{ 2^n \;\middle|\; n \in \mathbb{N} \wedge 2^n < 2^{S_\Phi(x)} \right\}\right| \\ & = & \left|\left\{ 2^0, 2^1, 2^2, \ldots, 2^{S_\Phi(x)-1} \right\}\right| \\ & = & S_\Phi(x) \end{array}$$

and

$$\begin{array}{rcl} S^{\mathcal{T}}_\Phi(x) & \overset{(4.6)}{=} & |\{ S_\Psi(y) \mid \Psi \in \mathcal{T} \wedge y \in X_\Psi \wedge S_\Psi(y) < S_\Phi(x) \}| \\ & \overset{(4.7)}{=} & |\{ n \in \mathbb{N} \mid n < S_\Phi(x) \}| \\ & = & |\{0, 1, 2, \ldots, S_\Phi(x) - 1\}| \\ & = & S_\Phi(x) \quad ; \end{array}$$

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

By a similar argument, $\{ T_\Psi(y) \mid \Psi \in \mathcal{T} \wedge y \in X_\Psi \} = \mathbb{N}$, whence $T^{\mathcal{T}} = T$.

Note that the inconsistency discussed in Sect. 4.5.1 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. 4.5.1 is not possible.

Properties of Normalization.

Theorem 22 (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.[25] □

Corollary 4. 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 22, we have that (i) ⇔ (i′), (ii) ⇔ (ii′) and (iii) ⇔ (iii′).

[25] 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* $A_{\Phi_i}(x_i) \in \mathbb{N}$. (Of course, there exist *infinite* sets P and Q that satisfy $P \subsetneq Q$, $|P| \geq |Q|$; let P be the set of primes and Q the set of rational numbers, for example.)

Proof. Theorem 22 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

$$\begin{array}{rcl} A^{\mathcal{C}}_{\Phi_1}(x_1) & \overset{(4.6)}{=} & |\{\, A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_{\Phi_1}(x_1) \,\}| \\ & \overset{\text{(ii)}}{=} & |\{\, A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_{\Phi_2}(x_2) \,\}| \\ & \overset{(4.6)}{=} & A^{\mathcal{C}}_{\Phi_2}(x_2) \ ; \end{array}$$

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 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

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

So (i′) $\Rightarrow$ (i), (ii′) $\Rightarrow$ (ii) and (iii′) $\Rightarrow$ (iii). □

Theorem 23 (idempotence)**.**

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

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

have the same cardinality (this is by the Schröder-Bernstein theorem (Theorem 5.1.2 of [106]), which is evocable since Corollary 4 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.'.

$$\begin{array}{rcl} A^{\mathcal{C}\mathcal{C}}_\Phi(x) & \overset{(4.6)}{=} & \left|\left\{\, A^{\mathcal{C}}_\Psi(y) \;\middle|\; A^{\mathcal{C}}_\Psi(y) < A^{\mathcal{C}}_\Phi(x) \,\right\}\right| \\ & \overset{\text{Cor. 4}}{=} & \left|\left\{\, A^{\mathcal{C}}_\Psi(y) \;\middle|\; A_\Psi(y) < A_\Phi(x) \,\right\}\right| \\ & \overset{\text{eq.}}{=} & |\{\, A_\Psi(y) \mid A_\Psi(y) < A_\Phi(x) \,\}| \\ & \overset{(4.6)}{=} & A^{\mathcal{C}}_\Phi(x) \ . \end{array}$$

(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 27, $\mathcal{C}$-normal. □

Definition 28. For $n \in \mathbb{N}$, let $[[n]] = \{\, i \in \mathbb{N} \mid i < n \,\}$. Extend this notation by letting $[[\infty]] = \mathbb{N}$. Hence, $[[0]] = \emptyset$, $[[1]] = \{0\}$, $[[2]] = \{0, 1\}$, …, $[[n]] = \{0, 1, 2, \ldots, n-1\}$, …, $[[\infty]] = \{0, 1, 2, \ldots\}$.

(The set-theoretically minded reader may wonder why we introduce $[[0]]$, $[[1]]$, $[[2]]$, …, $[[\infty]]$ when there already exist ordinals $0, 1, 2, \ldots, \omega$; we use the '$[[\cdot]]$' notation in order to retain the distinction between ordinals and cardinals.)

Lemma 14. Let X be any subset of $\mathbb{N}$. If, for every $i \in X$, $[[i]] \subseteq X$, 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, $X \cap \{m+1, m+2, m+3, \ldots\} = \emptyset$. Hence, $X = [[m+1]]$.

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

Theorem 24 (normal ⇔ 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 ⇒ $\mathcal{A} = [[n]]$'.) 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 14 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,

$$\begin{array}{rcl} i & = & A_\Phi(x) \\ & \overset{\text{hypothesis}}{=} & A^{\mathcal{C}}_\Phi(x) \\ & \overset{(4.6)}{=} & |\{\, A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_\Phi(x) \,\}| \\ & = & |\{\, A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < i \,\}| \quad . \end{array}$$

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 $x \in X_\Phi$.

$$\begin{array}{rcl} A^{\mathcal{C}}_\Phi(x) & \overset{(4.6)}{=} & |\{\, A_\Psi(y) \mid \Psi \in \mathcal{C} \wedge y \in X_\Psi \wedge A_\Psi(y) < A_\Phi(x) \,\}| \\ & \overset{\text{defn. of } \mathcal{A}}{=} & |\{\, j \in \mathcal{A} \mid j < A_\Phi(x) \,\}| \\ & \overset{\text{hypothesis}}{=} & |\{\, j \in [[n]] \mid j < A_\Phi(x) \,\}| \\ & \overset{A_\Phi(x) \in \mathcal{A}}{=} & |[[A_\Phi(x)]]| \\ & = & A_\Phi(x) \quad . \end{array}$$

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

4.5.3 Why Normalization?

As we see in Sect. 4.5.1, leaving unchecked (other than the restrictions described prior to the present section) 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 certain of those desirable properties for which such notions were introduced. This is further bolstered by the following intuition.

If we were to allow as a resource an *arbitrary* function with codomain $\mathbb{N} \cup \{\infty\}$ and that satisfies Blum's axioms, 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 admit only *normal* resources,[26] 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 specific choice of resource, and of any unit of measurement suggested thereby, and so, we suggest, fairer, resource-heterogeneous comparison becomes available, and much of the deceptive complexity behaviour alluded to in Sect. 4.5.1 is avoided.

4.5.4 Summary of Resource

When measuring the cost of performing a computation *by Turing machine* or similar, our choice of resources is guided by decades of collective experience and intuition, and by Blum's axioms (of which the necessity, for our purposes, is common sense, even if the sufficiency is dubious). In the context of *non-Turing* computation, however, exactly what one should admit as a resource is a much less studied, much more poorly understood problem; starting with Blum's axioms seems reasonable (moreover, not starting with them arguably seems unreasonable), but we see above that they alone are not enough. Accordingly, we apply further restriction, namely, that considered resources be *normal*.

For convenience of reference, we summarize now our restrictions on the notion of resource. In the present work, and when analyzing the complexity of unconventional computers, we advocate that a valid (commodity) resource

- be a function, dependent upon the choice of computer, that maps input values to natural numbers (or to ∞) (see Sect. 3.4.1);

[26] We certainly advocate this restriction, and summarize the accordingly modified notion of resource below—see Sect. 4.5.4.

- satisfy Blum's axioms[27] (see Sect. 3.4.1 *Blum's Axioms*); and
- be $\mathcal{C}$-normal (see Sect. 4.5.2 *Definition of Normalization*), where $\mathcal{C}$ is the appropriate encompassing class of computers.

These are necessary conditions for a resource to be 'valid', though are not between them sufficient[28]; a full *definition* of resource remains an open problem, which, we believe, will be solved[29] only by continual and ad hoc addition of restrictions prompted by inconsistencies encountered (much like that described in Sect. 4.5.1), and absent properties desired (such as the restrictions' not alone guaranteeing any desirable results concerning *resource constructibility*, by which we mean the obvious generalization of time/space constructibility; compare with the approach to constructibility of Kojiro Kobayashi [90], whereby resources can be 'counted down' by the computers using them), by practitioners of unconventional computation/complexity theory.

This insufficiency notwithstanding, we reiterate that such restriction precludes some deceptive complexity behaviour, such as 'dominance engineering' via application of quickly growing, isotone functions, 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 the present work.

Aside. As a final, contextual note on normalization, we comment that, although the original motivation for the concept lies in unconventional computing, with its many associated resources, the '$\{S, T\}$ versus $\{2^S, T\}$' inconsistency applies in the conventional case also—key is not that the resources be many, but that they be *compared*. Normalization is therefore of wider interest than the context of the present work may suggest. We intend to present in future work the chief ideas of normalization in a form of interest to the wider (conventional) computational-complexity community.

4.6 Discussion

4.6.1 Summary

We motivate in Sect. 4.1 the practice of comparing the overall efficiency of computers (both standard and non-). So as to allow such comparison, notably when the computers are unconventional, we define in Sect. 4.2 this chapter's focal concept, *dominance*, as a criterion for determining a resource's 'relevance' to a computation, whence we may define computations' *overall complexity*; in Sect. 4.3, we define and investigate the corresponding complexity classes.

[27] From here, recall, we get many standard results from [31]: resources as advocated here are a subset of those for which Blum's theorems hold.

[28] 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 $n \mapsto 2^n$, it does nothing to preclude *understatement* of a resource's importance via application of functions such as $n \mapsto \lceil \log n \rceil$—recall Remark 41. Another illustration is that transformations of resources may result in $\mathcal{C}$-normality (and may adhere to our other restrictions) *whilst not being order-preserving*.

[29] If, that is, the problem is solved at all. It may, in particular, be the case that there is a form of *incompleteness*, along the lines of 'all finite lists of restrictions upon resource admit undesirable dominance engineering'; cf. Arrow's theorem. We thank Cristian Calude for this suggestion, which is to be investigated in future work.

We note in Sect. 4.4 that with our novel approach to complexity theory comes a need to test whether standard complexity-theoretic techniques and tools can be used unmodified; the Gap Theorem [34, 125] provides us with an illustrative example.

Finally, in Sect. 4.5, we observe that, in order to avoid certain undesirable complexity behaviour, dominance demands a more specific notion of resource than is described in Chap. 3; we restrict the notion accordingly by defining *normality* and stipulating that resources be normal.

4.6.2 Comments

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.[30] Thus, much as the pre-ordering $\lesssim$ can be used to compare the respective time complexity of Turing machines (or similar) and thereby compare their overall efficiency, $\lesssim$ can be used also to compare the respective $\mathcal{R}$-*complexity* of *arbitrary-paradigm computers* and thereby compare their overall efficiency.

Furthermore, we have specified (in Definition 26) 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-heterogeneous 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.

Recall from Sect. 4.1.1 *Complexity: Problems versus Solution Methods* that the model-heterogeneity of our framework offers an immediate and important advantage: a *problem's* complexity, which is arguably 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 project 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—see Sect. 4.1.1 *Turing-Machine Benchmarks*.

Having developed the theoretical basis for our complexity framework, we turn our attention in Chap. 5 to some concrete case studies.

[30] We must warn against the overuse (or, rather, inappropriate use) of dominance: the notion is suitable for comparing computers' complexity, but not for eliminating resources from our initial consideration (as when considering which resources are of interest for given computational paradigms—see Sect. 3.6). For example, we recall that a Turing-machine computation's space usage is always bounded above by its run-time, and so T is $\{S, T\}$-dominant for such computers (more generally, if the consumption of a resource during any unit time period is bounded above by a constant, then time will dominate the resource); this is not to say, however, that there is not useful information to be gleaned by considering space complexity—during computers' *comparison*, the use of dominance, and, hence, the neglect of space and similar, is appropriate, but, during individual computers' *analysis*, non-dominant resources may well be of interest.

Chapter 5

Case Studies

5.1 Introduction

We introduce above the constituent concepts—resource, dominance, overall complexity, normalization, etc.—of our model-independent framework of complexity theory. In this chapter, we apply the notions to specific contentious and controversial case studies, on the formalization/resolution of which the approach to complexity expounded in the present work (and especially the *role of precision* and other non-standard resources in these disputes) may shed light. Speaking of shedding light, we begin with a consideration of the ray-tracing problem.

5.2 Ray-Tracing —Computing the Uncomputable?

5.2.1 Background

One paradigm of *optical computation*, discussed in [110], sees the values with which it computes (including intermediate values analogous to a Turing machine's mid-computation tape contents) encoded as spatial coordinates of rays of light, and sees the operations applied to these values implemented using, for example, reflection/refraction by a surface.

The paper considers the *ray-tracing problem*, which asks, given the set-up of an optical system and the initial position and direction of a ray of light, whether the ray reaches a given point; in particular, the problem (which is, in a sense, naturally and efficiently solved by such an optical system) is shown to be undecidable[1] by Turing machine, which suggests of the abstract model super-Turing power.

We note in defence of [110] that the authors are clear from the outset that they deal with a theoretical model that is based on the assumption that optics works in an idealized, geometric way (with no wave phenomena, etc.),[2] which

[1] Even, [110] shows, if the objects forming the optical system are finitely expressible (e.g., via rational quadratic/linear equations) may this undecidability still be present.

[2] For example, we read (in the description of the ray-tracing problem) that

> "[r]ays are assumed to have infinitesimal wavelength and are treated as lines

context justifies their not having considered the precision issues inherent in their systems. When such issues *are* considered—when the distinction is drawn between mathematical model and implemented computer (recall Sect. 3.2.3 *Mathematical/Physical Mismatch*)—, however, one may expect that the problem actually being solved by the system, as impaired by its physically imposed limitations, is no longer undecidable; formalization within our complexity framework of the situation allows us to demonstrate in Sect. 5.2.2 that this is the case.

5.2.2 Resolution

(We apologize for the ambiguous wording here. 'Resolution' in this section is in the sense of being 'of controversies', rather than in the optical-computing sense of being 'of light sensors or similar'.)

The ray-tracing methods described in [110] make use of encodings whereby the entire tape contents[3] are encoded as the angle of a single ray of light, with one bit of precision in this angle being used for every tape cell to which is written.

We consider the precision complexity of this process in the simple case where we merely *store* as an angle an n-bit value (which one may identify with n cells' contents) only to *retrieve* it. Suppose that the system is two-dimensional, and, in particular, that the angle θ that encodes our value is a *plane angle* in the range $[0, 2\pi)$ (where the unit is the radian, and where we deem the interval to be 'circular' in that 0 and 2π are coincident, and, more generally, arithmetic is performed modulo 2π).

The intention is that the input (i.e., stored) value $x = \sum_{i=0}^{n-1} 2^{-i-1} b_i$ (for bits b_i) in the range $[0, 1)$ be encoded as $\theta_x := 2\pi \left(x + 2^{-n-1}\right)$—this is central in the range of angles $2\pi y$ such that x and y agree in the n most significant bits.[4]

Suppose now that we have an additive input error of ϵ_i: an input value $x \in [0, 1)$ is encoded as an (implemented) angle $\theta'_x \in [\theta_x - \epsilon_i, \theta_x + \epsilon_i)$. Suppose,

---

with zero width. This implies that there is no diffraction caused by the wave nature of light. All surfaces are perfectly smooth and do not cause the scattering of rays upon reflection or refraction".

Later, we read of the paradigm that,

> "[t]heoretically, these optical systems can be viewed as general optical computing machines, if our constructions could be carried out with infinite precision, or perfect accuracy. However, these systems are not practical, since the above assumptions do not hold in the physical world. Specifically, since the wavelength of light is finite, the wave property of light, namely diffraction, makes the theory of geometrical optics fail at the wavelength level of distances".

[3] We implicitly suppose, then, that computations via ray-tracing methods are instantiations of intermediate Turing-machine implementations; this imbues the model with universality (in the Turing sense), but may increase complexity of some problems—recall from Sect. 2.2.1 Susan Stepney's thesis that it is more desirable to let computers function 'naturally' than to coerce them into instantiating logic gates or similar.

[4] We wish to be able to store and retrieve an n-bit value, which entails sufficient precision to set and measure θ_x correct to at fewest n bits. Given that we wish (in order to encode n bits) to be able to distinguish, given measurements of θ_x, 2^n different states, the pigeon-hole principle dictates that at fewest one such state is represented under the encoding by a set of angles of measure at most $2\pi 2^{-n} = 2^{1-n}\pi$; further, this bound is attained by, for example, the scheme whereby each n-bit value $x \in [0, 1)$ is encoded by $\theta_x \in \left[2\pi x, 2\pi \left(x + 2^{-n}\right)\right)$, an interval of measure $2^{1-n}\pi$. This is implementable with our encoding.

similarly, that we have an additive output error of ϵ_o: angle θ'_x is measured as an angle in $[\theta'_x - \epsilon_o, \theta'_x + \epsilon_o)$. A measurement of an encoding of x, then, will be a value $m \in [\theta_x - \epsilon, \theta_x + \epsilon)$, where $\epsilon = \epsilon_i + \epsilon_o$; this is decoded (via $m \mapsto \frac{m}{2\pi} - 2^{-n-1}$) as $x' \in \left[x - \frac{\epsilon}{2\pi}, x + \frac{\epsilon}{2\pi}\right)$.

For x and x' to agree to n bits (and given that x is encoded centrally in its range, as above), we need that $x' \in \left[x - 2^{-n-1}, x + 2^{-n-1}\right)$, which entails that $\frac{\epsilon}{2\pi} \leq 2^{-n-1}$—i.e., that $\epsilon \leq 2^{-n}\pi$. So (recalling the notation of Sect. 3.3.3)

$$\mathcal{E}(x) = \left\{ (\epsilon_i, \epsilon_o) \in \mathbb{R}^2 \mid \epsilon_i, \epsilon_o \geq 0 \wedge \epsilon_i + \epsilon_o \leq 2^{-n}\pi \right\}$$

(see Fig. 5.1), $\mathcal{V}(x) = 2^{-2n-1}\pi^2$ and $P(x) = 2^{2n+1}\pi^{-2}$.

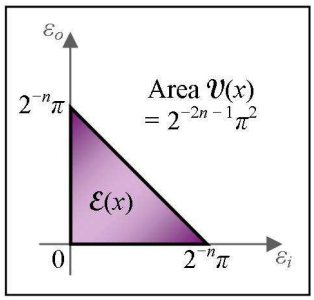

Figure 5.1: The set $\mathcal{E}(x)$ of errors corrigible for the process of storage and retrieval of an n-bit value.

Thus, the precision complexity of the process of merely storing and retrieving a value is *exponential* in n (i.e., exponential in the number of tape cells written to by an intermediate Turing-machine implementation).

Of course, this increasing (let alone exponentially increasing) precision complexity has the following consequence: if the precision available to a user of such a system as described in [110] is a priori bounded (e.g., by technological factors), *which, in practice, it necessarily will be*, then the size, too, of problem instance that can be processed by the system is bounded. The system, then, solves not some undecidable problem in its entirety, but rather a proper subproblem (admitting only bounded input instances, and concerning only physically realizable optical configurations rather than those with coordinates drawn arbitrarily from the continuum of real numbers or even from the countable infinity of rational numbers); this subproblem may well be computable—*efficiently* computable, even—by Turing machine. This is our resolution of the controversy surrounding the findings of [110].

Aside. A more striking and sadly lacking resolution would see required precision being shown to be *infinite*, not merely in the limit, but also for some finite input size; this is simply not the case, however (at least, not for the *commodity* resource of precision; the *manufacturing* resource is a different matter—see the following aside). Whereas an exponentially increasing precision is required to implement the methods described in [110], it is at least a *finite* precision—the method is precluded not *in principle*, but only *technologically*, by issues concerning precision. We recall in particular that [110] discusses the computation of the *uncomputable* via such methods; one might, given the extended Church-Turing thesis, therefore expect preclusion *in principle*, which, as explained, is not the case.

The above aside notwithstanding, however, we reiterate that, concretely and technologically, precision bounds the system's processable input sizes; additionally, we note that, abstractly and fundamentally, precision (and, in particular, its increasing) renders problematic, when input values are sufficiently large, certain seemingly innocuous assumptions (that diffraction is not present, etc.). The idealized model of optical computation does indeed seem capable of computing the uncomputable, but real-life implementations necessarily suffer because of the 'subtle' differences (that become less subtle for large input) between the theoretical and practical models,[5] differences that arise due to apparently harmless assumptions that quickly become problematic in the presence of exponential precision complexity. The extended Church-Turing thesis is not, we suggest, violated, simply because the proposed optical system is not a valid, real-world computer (by virtue of unrealistic assumptions regarding precision).

Aside. In line with the discussion of Sect. 3.2.4 *Justification of Choice*, we consider above the role of precision as a *commodity* resource; the systems of [110] also incur a significant *manufacturing* precision cost, however. Theorem 5.7 of [110] relies on the introduction to an optical system of irrational numbers, which introduction is achieved via precise construction of lenses/mirrors with specific, irrational focal lengths (a rational approximation of which, however accurate, not having the desired effect). The precision required of such systems is, therefore, prohibitive as not only a commodity, but also a manufacturing resource (recall Sect. 3.2.1 *Manufacturing Costs*).

We emphasize that it is not the aim of the authors of [110] to assess *practically implemented instances* of their systems; by their own admission, the authors are working relative to explicitly stated assumptions, and their findings are, in that context, perfectly correct. What we formalize here is that these assumptions, by virtue of their precision implications, render the considered systems unsuitable as real-world computers, and consequently render the findings of [110] irrelevant to the question of whether hypercomputation is *possible* in any real sense.

5.3 An Adiabatic Solution to Hilbert's Tenth Problem

5.3.1 Background

The quantum adiabatic theorem (of which an elementary proof is supplied in [6]) says, essentially, that if a quantum system of a certain type evolves sufficiently slowly, then the final energy state will correspond to the initial energy state. In particular, if the system begins in the *ground* state and evolves over sufficient time, then it will end in the ground state. Though the proof of the theorem given in [6] is not, by some decades, the first, its presentation is worthwhile due to its being elementary and aimed at computer scientists rather than physicists[6]; that computer scientists are conversant with the theorem is, in turn, desirable—and

[5] Recall also Sect. 3.2.3 *Mathematical/Physical Mismatch*.

[6] The approach, further, offers geometric intuition as to why sufficiently slow evolution maintains the ground state.

that the theorem is to the present context relevant—because the phenomenon that the theorem guarantees can be used as the basis for *computation*.

Reference [64] describes production of an initial ground state that encodes a given instance of a problem (the example considered is 3-SAT) and a system that evolves such that the final ground state encodes the answer to the problem (i.e., 'yes' if and only if the instance is satisfiable). After verifying the hypotheses of the theorem (notably that there is, throughout the evolution, a non-zero gap between the 'zeroth'—i.e., ground—and first energy states), the theorem gives that, if evolution is sufficiently slow, then our final state encodes the sought answer. The system certainly seems to tackle the 3-SAT problem, then; what of its time complexity—how does the theorem's "sufficiently slow" scale?

Unfortunately, by the admission of the authors of [64], the subset of instances for which this time complexity can be readily analyzed (and for which it is found, for what it is worth, to be polynomial) succumb to efficient—i.e., polynomial-time—processing by Turing machine; the system, then, is an important proof of concept of quantum-adiabatic computation, rather than a necessarily efficient means of solving 3-SAT for general instances.[7]

In [87,88], Tien Kieu offers a quantum-adiabatic algorithm that purports to solve *Hilbert's Tenth problem*. The problem, posed in [77], asks whether there exists an integer solution to a given Diophantine equation (i.e., a multivariate polynomial with integer coefficients); the problem is known (via reductions in each direction—see [55]) to be equivalent to the Halting problem, which is, in the Turing-machine realm,[8] undecidable (see [127]). The (Turing-) undecidability of the problem understandably renders controversial the claims of [87, 88]: claimed is not merely an improvement in *complexity*—that quantum-adiabatic computers are, say, faster than Turing machines—, but an improvement in *computability*—that, regardless of efficiency, Turing machines can compute strictly fewer[9] functions than can quantum-adiabatic computers.[10]

Despite Kieu's claims' not having been widely accepted by the scientific community, neither has a formal and precise refutation been given; we suggest that this is because the resources (especially precision) appropriate to a complexity analysis of this particular adiabatic quantum system have not been considered, and, hence, that the notions of the present project can be used to formalize the reason for which the system does not in fact violate the extended Church-Turing thesis (which violation is, after all, the essence of the controversy).

---

[7] The wider quantum-adiabatic computation technique of [64] (as opposed to the specific 3-SAT case) is generalized in [63]. Whereas the former reference assumes linear interpolation between initial and final Hamiltonians (i.e., the form taken is $(1-s)\,H_{\mathrm{i}} + sH_{\mathrm{f}}$, where $s \in [0,1]$ and where H_{i} stands for the initial, and H_{f} the final, Hamiltonian), the latter concerns different paths between H_{i} and H_{f}; in some cases, this variation of paths results in polynomial time complexity where the linear path would have entailed exponential run-time.

[8] The standard proof of the problem's undecidability relies on Turing machines' being only countably many, and does not preclude probabilistic/quantum/etc. solution.

[9] Fewer, that is, in the sense of being a strict subset, rather than of having a smaller cardinality; recall Footnote 25 of Chap. 4.

[10] Note that quantum-adiabatic systems are essentially *probabilistic*, guaranteeing the correct answer to a computation only in the limit, where computation time becomes infinite; the method of [87, 88] in particular entails the less-than-ideal practice of executing for longer and longer durations until a state (which we infer to be the sought ground state) emerges with probability at least $\frac{1}{2}$. A facile 'resolution' of the controversy surrounding the system of [87, 88], then, is that Hilbert's undecidable problem is not, in fact, being *solved* at all; we wish, however, for more insight than this.

5.3.2 Resolution

We seek to apply our framework of complexity to resolve the controversial issues surrounding the system of [87, 88]. We claim that the issue is one of *precision*; this is argued informally in [79], where we read that

- "[Kieu] is charging a simple and finite physical system (a harmonic oscillator) with an *infinite amount of data*. But this is a fundamental point: the basis of Kieu's claim is that an infinite amount of information can be compressed into a finite physical system";

and

- "Kieu's infinite data storage needs *zero error* to work, and really does depend on setting up and maintaining an infinitely precise system".

Most tellingly, we read in [79], of a certain process used in Kieu's system, that

> "[t]he first step is to encode ... the 2 as 2.00000... and the 1 by 1.00000.... The slightest error in this transcription will (for sufficiently large values of M and N) completely wreck the calculation of the polynomial and invalidate the search for its minimum value".

That this is indeed so, and that this corresponds to monotonically—and, we suspect, exponentially—increasing precision complexity (in our formal sense—recall Sect. 3.3.3) is confirmed by an examination of the description, in Sect. II of [88], of the process. The intuition of [79], then, is captured by our resource of precision.

Just as with the precision complexity of the systems of Sect. 5.2, the increasing precision complexity suffered by Kieu's system has the following consequence: if user precision is a priori bounded, *which, in practice, it necessarily will be*, then the workable problem-instance size is also bounded. The problem solved by the system is, analogously to the situation of Sect. 5.2, a *proper subproblem* of Hilbert's Tenth, and may well be efficiently computable by Turing machine. Thus we dispel this, as the above, controversy.

5.4 Cabello versus Meyer—the Impact of Finite Precision on the Kochen-Specker Theorem

5.4.1 Background

In Sect. 5.4, we consider the impact upon the Kochen-Specker theorem (see Sect. 5.4.1 *Kochen-Specker Theorem* and [89]) of finite precision. Specifically, it is not clear whether 'finite-precision measurement nullifies the Kochen-Specker theorem'—Meyer, in the title of [96], claims exactly that, whereas Cabello, in the title of [44], negates the statement. In Sect. 5.4.2, we take the first steps toward formalization and resolution (relative, at least, to the assumptions of our framework) of the dispute.

We outline now some preliminary notions.

Hidden-Variable Theories.

These theories[11] were put forward (and held by only a minority) to account for the probabilistic nature of certain quantum phenomena; the suggestion is essentially that quantum mechanics is, behind the scenes, deterministic, and that we simply have yet to derive its full laws. This view is in part motivated by the belief, expressed in [61], that a *complete* physical theory must be *deterministic*. Such theories say, then, that behind the probabilistic effects of quantum physics lie definite, actual states, with the probabilism merely being a manifestation of our ignorance of these states.

Bell's Theorem.

This theorem, derived in [14], suggests that *local* hidden variables alone are insufficient to account for quantum-mechanical phenomena. As a notable example, EPR pairs (see [61]) feature disparate correlations that, if reliant on hidden variables, necessitate *non-local* ones. This illustrates the derivation of necessary conditions on hidden-variable theories that describe quantum mechanics, of which derivation the Kochen-Specker theorem provides another example.

Kochen-Specker Theorem.

This theorem, given in [89], constrains the hidden-variable theories that may describe quantum mechanics. The theorem demonstrates an inconsistency between two assumptions about hidden variables:

1. that they have definite values at any given time, and
2. that these values are intrinsic and independent of the choice of measuring device/method (or, equivalently, of what other measurements are performed simultaneously).

The method of proof, roughly speaking, is to formalize systems of hidden variables as functions that embed quantum theory into a (deterministic) classical theory, whence the derivation of necessary conditions for these functions leads to their non-existence, provided that two desirable conditions[12] hold.

Meyer [96] and Cabello [44].

The suggestion of [96] is that the feature of finite-precision measurement relevant here is its insufficiency to distinguish a dense subset from its closure; it is on this feature, rather than on the reasons[13] for which it is indeed the relevant one, that we focus here. Ultimately, [96] concludes that the availability of only

[11] Examples are the pilot-wave theory of [56] and its updated form, the Bohm interpretation, of [32].

[12] These conditions are, essentially, that statistical predictions concerning a system reflect the hidden variables' values (condition (1) of [89]), and that the algebraic structure induced by virtue of quantum observables' commensurability be respected by the embedding (condition (4)).

[13] The reasons, which concern colourings of certain triples of points, are beyond the scope of the present work. We hint only that colourings with specific properties witness the existence of certain hidden-variable theories (hence the link with the Kochen-Specker theorem), and that each triple is arbitrarily close to a colourable one (hence the link with sets' density).

finite-precision measurement renders the Kochen-Specker theorem unsuitable as a means with which to demonstrate that there cannot exist the hidden-variable theories (consistent with the statistical predictions of quantum mechanics) of the type purportedly precluded by the theorem.[14]

However, [44] claims that the argument of [96] relies on exhibiting rogue triples, which, colourable though they may be, are not themselves consistent (in a certain desirable way) with quantum mechanics.[15] The implication is that, even given only finite precision, the Kochen-Specker theorem is valid.

Aside. As additional motivation for our attempt to formalize the precision issues involved in this dispute—in particular, as a suggestion that the approach is promising—, we recall from [44] that

> "Ax and Kochen [in unpublished work] have argued that the study of the effect of finite precision measurements on the KS theorem requires a different formalization which is still missing".

We take now the first steps in applying such a formalization.

5.4.2 Resolution—First Steps

As a partial resolution of this controversy, we turn to the above-mentioned feature of finite-precision measurement, with a view to its formalization; the indirect implications of this formalization are beyond the scope of the present project. We recall the above suggestion of [96]: that relevant to the present work is that finite-precision measurement is *insufficient to distinguish a dense subset from its closure*. We verify now that this feature is indeed true under our notion of precision.

Let A ($S^2 \cap \mathbb{Q}^3$ is the specific set considered in [96]) be a dense subset of S^2 (i.e., the unit sphere in $\mathbb{R}^3$). By density of A, we have that the closure of A is S^2; i.e., that

$$A, \text{ in union with its limit points}^{16}, \text{ is } S^2.$$

Now, given a point $a \in S^2$ and granted only finite precision, we claim that one cannot determine whether $a \in A$. Since a is in the closure S^2 of A, either $a \in A$ or a is a limit point of A; if the latter, then $(\forall \epsilon > 0)\, [B(a, \epsilon) \setminus \{a\}] \cap A \neq \emptyset$. If, therefore, $a \notin A$ but a is a limit point of A, then $(\forall \epsilon > 0)\ (\exists a' \in A)\, d(a, a') < \epsilon$. Thus, finite precision (with respect to metric d, of which choice does not matter—see Prop. 20) implies that '$a \in A$?' is unanswerable.

Definition 29. We say that a measuring process has $\frac{1}{\epsilon}$*-precision* (and write 'infinite precision' for '∞-precision') with respect to a metric d if a point x is necessarily measured as some point y such that $d(x, y) \leq \epsilon$.

[14] Of course, it is the *proof method*, rather than the *conclusion* that [96] doubts; the suggestion is not that these hidden-variable theories *do* exist, but that Kochen-Specker *does not show that they do not*.

[15] R. Clifton claims in a private communication with Cabello that this claim, in turn, is based upon an incorrect assumption...

[16] Limit points are those points, l, such that, for all positive ϵ, $[B(l, \epsilon) \setminus \{l\}] \cap A \neq \emptyset$, where $B(l, \epsilon)$ is the open ϵ-ball around l (with respect to some metric, of which choice is not important—see Prop. 20).

(As motivation for this definition, note that precision complexity is proportional to $\frac{1}{\epsilon}$, which is infinite if and only if $\epsilon = 0$.)

Proposition 20. Let d_1 and d_2 be arbitrary metrics. A process has infinite precision with respect to d_1 if and only if it has infinite precision with respect to d_2.

Proof. Since d_i are metrics, we have that, for all x and y,

$$d_i(x, y) \geq 0 \tag{5.1}$$

and

$$d_i(x, y) = 0 \Leftrightarrow x = y \ . \tag{5.2}$$

Suppose infinite precision with respect to d_1. Then x is measured as some y such that $d_1(x, y) \leq 0$. By $(5.1)_{i=1}$, $d_1(x, y) = 0$; so, by $(5.2)_{i=1}$, $x = y$. But then, by $(5.2)_{i=2}$, $d_2(x, y) = 0$, so, a fortiori, $d_2(x, y) \leq 0$. That is, we have infinite precision with respect to d_2.

By swapping the roles of d_1 and d_2, we obtain the converse. □

We have shown that, according to our definition of precision, the hypothesis (concerning subsets' density within sets) of Meyer's argument [96] is satisfied, whence its evocation and conclusion are as valid (which, [44] suggests, is *in*valid) in our framework as they were in the context of [96]. Of course, the bulk of the controversy remains, but we have at least set the stage for its formalization and resolution.

5.5 Future Work

The previous lack of a rigorous means of treating the role of precision in computation, especially unconventional computation, has as a result many instances in the literature of incompletely understood situations that may well benefit from formalization in our framework. We defer such formalization to future work, but mention in passing as candidates the following.

- Consideration is made in [84] of the way in which undecidability leads to uncertainty about physical systems. However, we note that undecidability is defined in terms of *Turing machines*, and so ask (1) whether this restricted view affects the conclusions, and (2) whether the uncertainty is not captured by the notion of 'impossibilities' (such as measuring with infinite precision). The apparent answer of [84] to (1) is negative:

  > "[i]n general, as we will explain, the undecidability principle applies to any physical system [including continuous-time/space systems]";

  however, the extension to the continuous case proceeds essentially by embedding and appealing to discrete undecidability, whereas certain non-Turing systems in some sense (which, we suggest, is the specific candidate for formalization within our framework) decide their 'undecidable' behaviour. For (2), we conjecture that there is such a link; a particular direction for future study here is to investigate the boundary, in terms of implications concerning uncertainty, between finite and infinite precision.

- Reference [91] demonstrates that physical theories, including the striking example of Newtonian mechanics, may give rise to uncomputable functions. A topic for future study, then, is whether *physically realistic* restriction of non-standard complexity measures such as precision resolves this situation, in that uncomputable functions become unavailable under such restriction.

- We read in [35] that,

  > "[a]lthough it has been shown that some continuous time models exhibit super Turing power, these results rely on the use of an infinite amount of resources such as time, space, precision, or energy. In general, it is believed that "reasonable" continuous time models cannot compute beyond Turing machines".

  This sentiment is a clear candidate for formalization within our framework (though we conjecture that, once so formalized, the claim may be too general to be proven in one swoop, relying instead upon properties of individual computational models or similar).

- Though regarded by the scientific community with less seriousness than the controversial publications mentioned in Sects. 5.2, 5.3 and 5.4, [42] nonetheless contains an argument (very roughly, 'efficient' soap-bubble methods exist for NP-complete problems, therefore P = NP) that may succumb to formalization (and, thence, refutation) within our framework, as is suggested by Footnote 8 of Chap. 3.[17]

- The resolution begun in Sect. 5.4 has, as we comment above, yet to be completed.

5.6 Discussion

5.6.1 General Features of Resolution

We see above, common to Sects. 5.2 and 5.3, a theme: that these controversial systems are plagued by increasing precision complexity, which has as a consequence an upper bound on the size of input value that the system can successfully process. Whereas a snappier resolution along the lines of 'the system requires *infinite* precision and is therefore unviable' is not forthcoming, the fact that precision is at least *increasing* (finiteness notwithstanding) gives us as much resolution as we need. Given a practical limit[18] on precision, the problem solved by the system ceases to be undecidable, and the controversy, for practically implemented systems, is dispelled; only from a theoretical and abstract point of view may the systems well demonstrate (for what it is worth)

[17] Unfortunately, just as Meyer's argument (that the Kochen-Specker theorem does not establish non-existence of certain hidden-variable theories) does not itself establish *existence* of these hidden-variable theories, so our suggestion (that [42] does not establish P = NP) does not itself establish P ≠ NP...

[18] This may, for example, be imposed by technological constraints, or by required precision's becoming sufficiently great that effects (wave phenomena, for example) 'abstracted out' of a mathematical model cease to be negligible.

that their respective models—optical and quantum-adiabatic—, including unrealizable and often tacit assumptions about available precision, are strictly more powerful than the Turing machine.

5.6.2 Extent of Resolution

We consider now the extent to which the above controversies/disputes are resolved by the discussion of Chap. 5.

We introduce above a framework in which certain of the relevant issues may be (and above are) formalized; this enables pertinent questions—ultimately, we hope, 'is the disputed claim true or not?'—to be formulated precisely and answered definitively. However, it must be noted that the question actually answered is, 'is the disputed claim true or not *according to the framework and the assumptions implicit in its definition*?'; it is trivial, moreover, to suggest some framework that merely *deems* a disputed claim to be valid, or to be invalid.

The important factor, then, is the success with which our *formal framework* of resources, complexity, dominance, etc. captures our *intuitive understanding* of computational resource and related notions. The question of the extent of this success is not susceptible to rigorous proof, being (much like the Church-Turing thesis) a question of equivalence between a formal and an intuitive notion; rather, the equivalence must be borne out by evidently sensible choice, and sustained successful use, of the framework's definition—or else the equivalence fails. The former test has hopefully been shown, in the discussion and justification of our definitions, to have been passed (for now...); the latter is necessarily ongoing.

Chapter 6

Discussion

6.1 Other Applications

By design and construction, our framework of unconventional-computer complexity theory allows analysis of arbitrary-model systems' complexity with respect to arbitrary resources. For instance, we see in Chap. 5 the application of the framework to various disputes concerning systems' complexity (typically precision complexity); such application, then, is an example of the framework's 'intended use'.

Additionally, our defining as part of the framework such notions as *resource* and *complexity* has the (not undesirable) byproduct of suggesting analogues of existing concepts *defined in terms of* these notions. For example, much of *cryptography* is based upon the notion of functions' computation's being 'difficult' (prohibitively so for an eavesdropper, say); each formulation of 'difficulty', then, induces a corresponding concept of cryptography.

By their 'side-issue' nature, these analogues induced by our definitions are not the focus of the present dissertation, but we mention them now in passing and defer further detail to future work.

6.1.1 Cryptography

A investigation of particular interest for future work is into the cryptographic applications of the notions of this project. We mention above that with our formulation of complexity (and, hence, of difficulty) comes a corresponding formulation of many cryptographic concepts. We outline now two illustrative such concepts.

Confidence in a Problem's Hardness.

Public-key cryptography relies heavily upon our finding difficult certain tasks, such as factorization (see Sect. 2.1.1) or the computation of discrete logarithms (see, e.g., [58]). Use of the framework here described, which considers complexity in a more general sense than is typically the case, may increase our confidence that such tasks are indeed difficult, in that not only efficient Turing, but also efficient non-Turing, solution is unlikely—we see, for example, in Chap. 2 that

even 'apparently efficient'[1] (non-Turing) factorization systems may in fact be impeded by complexity behaviour relating to precision or similar.

Non-Turing Analogues of Trapdoor/One-Way Functions.

Another question for future work asks whether unconventional-computer analogues of *trapdoor functions* (which are easy to compute given some 'trapdoor' information, but computationally infeasible without—see [58]), *one-way functions* (which are easy to compute, but infeasible to invert, at least for a high proportion of image points—see [132]), etc. exist.[2] Of course, given that one does not generally know to which paradigm an adversary's computer conforms, the relevant question as far as cryptographic applications are concerned is whether there exist functions that are trapdoor, one way, or similar not just for one specific computational paradigm, but for all.

As a final cryptographic application for suggested future work, we note that, given a model-independent formulation of one-way functions, it is a small and natural step to investigate the corresponding notion of *pseudo-random number generation* (see, for example, [132]).

6.1.2 Kolmogorov Complexity

Another non-Turing analogue of a Turing-machine-based concept that presents itself as a candidate for future study within our framework is *Kolmogorov complexity* (see [48, 93]).

A string's Kolmogorov complexity can be defined, roughly, as the minimal size of a description of a Turing machine that produces the string.[3] We suggest for future study the analogues in which "a Turing machine" in this description is replaced with, for example, "an analogue computer", or even "a general computing system".

We recall from Sect. 3.8 the space/time trade-off (during values' storage and retrieval) suggested by the ideas behind Kolmogorov complexity: in essence, a *compressed* form of a string requires less storage space, but more read/write time (in which to perform decompression/compression), than the string in full. From different notions of compression, Kolmogorov complexity, etc. (such as the analogues described above) arise different 'balances' between the resources—time and space—involved in this trade-off; this offers another topic for future study.

[1] That this efficiency is apparent stems from the commonplace but, we argue above, counterproductive belief that standard complexity analysis suffices for non-standard computers.

[2] Such notions as trapdoor and one-way function that are defined in terms of computations' 'easiness' or 'infeasibility' are technology-dependent: as computers become more powerful, for example via either quantitative changes to numbers of transistors, etc. or qualitative changes to choice of computational paradigm, then more functions are rendered easy, and fewer infeasible, to compute. This suggests that model-independence, such as is exhibited in our framework, is a desirable property when investigating these cryptographic notions.

[3] As an aside, we note that [7] presents the interesting view that the large part of science that seeks to derive underlying laws and patterns from observed data can be viewed as algorithmic compression—which notion is very closely related to Kolmogorov complexity—of the data.

6.1.3 Cardinality and Set Theory

There is a fundamental connection between *precision* and the *dimension* of certain mathematical spaces. This is evident in proofs that the cardinality of $\mathbb{R}^n$ (for $n \in \mathbb{N} \setminus \{0\}$) is independent of the dimension n—there is a continuum of elements regardless; the obvious proof is essentially by *interleaving* decimal expansions: an n-tuple of real numbers with expansions[4]

$$a_{k_i,i}a_{k_i-1,i}a_{k_i-2,i}\ldots a_{1,i}.b_{1,i}b_{2,i}b_{3,i}\ldots$$

($i \in \{1,2,3,\ldots,n\}$) can be interleaved so as to map in an injective way to a single real number with expansion

$$\begin{gathered}(a_{k,n}\ldots a_{k,1})\,(a_{k-1,n}\ldots a_{k-1,1})\,(a_{k-2,n}\ldots a_{k-2,1})\ldots(a_{1,n}\ldots a_{1,1})\\ .\,(b_{1,1}\ldots b_{1,n})\,(b_{2,1}\ldots b_{2,n})\,(b_{3,1}\ldots b_{3,n})\ldots\ ,\end{gathered}$$

where $k = \max\{k_1,k_2,k_3,\ldots,k_n\}$ and $a_{j,i} = 0$ if $j > k_i$ (we do not account in this description for the *signs* of the real numbers, though these can be encoded merely as an additional digit inserted next to the decimal point, for example). (Of course, there exist injective maps—e.g., $x \mapsto (x,0,0,0,\ldots,0)$—in the other direction, from $\mathbb{R}$ to $\mathbb{R}^n$, whence these sets' equipollence by Schröder-Bernstein (Theorem 5.1.2 of [106])).

However, this and similar proofs rely on *unbounded precision*, in that the injectivity of interleaving follows from the distinctness of arbitrarily close real numbers. One may question, then, whether the theory (of cardinality, sets, etc.) that arises from a restriction to *bounded* precision is worthy of study (and whether, indeed, it differs at all from the existing theory of a discrete set such as the integers—note that the dimension/precision link described above, recast in the context of bounded precision, becomes essentially the issue of whether $|\mathbb{Z}^n|$ is independent of n, which it is).

We propose for future study, then, the fragment of set theory where constructions in proofs are permitted to use only 'achievable', bounded-precision operations.

6.2 Fundamental Questions

We briefly mention now some fundamental questions concerning the nature of computation itself; we suggest that these questions may succumb (at least in part) to analysis in the framework of the present project, though, as with much of Sect. 6.1, we defer the details pertaining to some of the questions to future work.

6.2.1 Inherence of Complexity in Problems

The question here is

*is complexity inherent in **problems**, or is it an artefact of our choice of **computer/paradigm**?*

[4] In these expansions, the a's are digits in the integer part of a real number, and the b's digits in its decimal part; sequential composition denotes concatenation of digits, and grouping via bracketing is for clarity rather than multiplication or similar. For example, expansions $a_2a_1.b_1$ and $(a_2a_1).b_1$ both have value $10^1 \times a_2 + 10^0 \times a_1 + 10^{-1} \times b_1$.

That is, are problems inherently easy/difficult, regardless of the computer or model employed in their solution, or does at least some of the complexity arise from our approaching the problems in certain ways? (Cf. Sect. 3.2.3.)

We conjecture that the spirit of the answer is that complexity is indeed inherent in problems: we find above repeatedly that, when a problem thought to be difficult for Turing machines is approached via other models, then it remains difficult, not necessarily by virtue of prohibitive time or space complexity, but often because of concerns of precision or similar. (That this is only the *spirit* of the answer reflects that, to a certain extent, choice of computer/model *can* affect complexity in that, even for easy problems, it is possible deliberately to choose a poor computer; we feel that the spirit of the answer should discount such contrary choice.)

As further evidence for this conjecture, note (e.g., from Sect. 4.2 of [37]) that the class P is *robust*, in that the class may be equivalently formulated in terms of Turing machines (of which exact details, such as number of tapes, are unimportant), calculable statements of logics, etc. That each formulation gives rise to the same class suggests a fundamental nature—more fundamental, we suggest, than choice of computational device/model—of the property of a problem's being efficiently soluble.

Aside. We note in passing that a potentially relevant factor in determining whether a given problem is inherently difficult is whether the problem is chiefly *combinatorial* or *numerical*. Often our observation above of exponential precision complexity arises because of the numerical nature of the problems considered, whereas one may expect discrete, combinatorial (e.g., search) problems to be better suited to the imprecise though highly parallel architectures offered by some unconventional paradigms; nonetheless the above conjecture seems to hold: even promising unconventional systems for combinatorial tasks often transpire to have prohibitive precision complexity, for example.

6.2.2 Model- and Resource-Heterogeneous Comparison

We answer affirmatively above the important question,

> *can meaningful comparisons be made between computers conforming to* ***different paradigms*** *and/or with respect to* ***different resources****?*

Specifically, much of Chap. 4 deals with exactly this issue.

6.2.3 Source of Systems' True Complexity

We may pose the question,

> *where does the* ***true complexity*** *of* $\star$ *computers lie?*

where we may take '$\star$' to be one of 'quantum', 'chemical', 'analogue', etc., or even 'arbitrary-model'.

We discuss this question for specific models in Sect. 3.6 and for the arbitrary-model case in Sect. 3.5; more fundamentally, we discuss in Sect. 3.2.3 various interpretations of the term 'complexity' as it appears in this question.

6.2.4 Computer/Environment Boundary

The question,

what delimits a ***computer*** *from its* ***environment****?*

is particularly relevant (and difficult) in the context of unconventional computation, since certain such computers defer processing tasks to their environment, as we now describe.

Turing machines present an abstraction of *self-contained* computers, within which the entirety of a calculation is performed: apart from the provision of input/output means, a Turing machine has no contact with anything external to itself. In contrast, *analogue* computation, for example, often exploits the fact that systems sit in a universe that itself calculates (according to physical laws)—one may defer some processing, then, to the environment. Whereas a Turing machine neither knows nor cares about the presence of gravity, for instance, an unconventional computer may exploit such, for example to square numbers by measuring the (output) distance that a mass falls in a manipulable (input) amount of time.

Of course, during complexity analysis, we wish to prevent our computers 'cheating' by unrealistically deferring computation to their environment (for else our impression of their complexity, and that of the problem being solved, is an underestimation); this renders the question an important one.

6.2.5 Underlying, Fundamental Resource

As a final question in this section, we ask

does there exist an underlying, ***fundamental resource****,*

of which other resources are facets? This is the topic of study in Sect. 3.8, to which section (and to future work) we defer further detail.

6.3 Conclusion

6.3.1 Summary

We summarize now the above content of this dissertation.

Chapter 1—*Introduction.*

We begin the dissertation with some introductory comments in Sect. 1.1, and by briefly recapping some preliminary notions—namely, complexity theory, computation, Turing machines, the Halting problem, the original and extended Church-Turing theses, unconventional computation, resource, Blum's axioms, complexity functions, asymptotic notation and natural numbers—in Sect. 1.2; in Sect .1.3, we outline the dissertation.

Chapter 2—*Motivation.*

In Sect. 2.1, we recall the problem of *factorization*, and note the lack of an efficient (polynomial-time) solution conforming to a standard, Turing-machine-like computational model. We introduce in Sects. 2.2 and 2.3 two analogue systems that solve the problem of factorization in *polynomial* time and space, but show that the systems' *precision complexity* (which concept we informally outline in Sect. 2.2) is *exponential*. As we note in Sect. 2.4, this suggests that, in order successfully to analyze the complexity of *unconventional systems*, one may well need to consider *unconventional resources*; this is the chief motivation for our implementation and investigation of the present project's model-independent approach to complexity theory, which we describe in Chaps. 3 and 4.

Chapter 3—*Resource.*

We begin, in Chap. 3, to describe our framework of model-independent complexity theory. In Sect. 3.1, we reiterate the need to consider unconventional resources, but note that they are very often overlooked. In Sect. 3.2, we make clearer what we mean by 'resource', outlining our chosen notion of *commodity resource*. Before formalizing the notion, we discuss in Sect. 3.3 an illustrative example: *precision*. In Sect. 3.4, we return to, and begin to formalize, the concept of commodity resource, going on to consider specific examples, some that pertain to all computational paradigms (Sect. 3.5), some of less general applicability (Sect. 3.6). In Sect. 3.7, we contrast traditional resources—time and space—with their unconventional counterparts. Finally, in Sect. 3.8, we consider issues surrounding the existence of an underlying, fundamental resource, of which all others are facets.

Chapter 4—*Dominance.*

In Sect. 4.1, we motivate the *comparison* of computers' efficiency, noting that the Turing-machine case is well understood, but suggesting that the unconventional-computer case is lacking. We address this by introducing *dominance* and $\mathcal{R}$-*complexity* (in Sect, 4.2), and the corresponding complexity classes (in Sect. 4.3, where we also prove theorems concerning the classes). In Sect. 4.4, we verify that the traditional *Gap Theorem* still holds in our framework, as an illustrative instance of confirmation that complexity-theoretic tools are still available. We note in Sect. 4.5 that resources as described in Chap. 3 are sufficiently permissive to allow, when using dominance and related notions, undesirable complexity behaviour; we introduce a further restriction—*normality*—to our concept of resource as a step towards addressing this. The ideas introduced in Chaps. 3 and 4 form our model-independent framework of complexity.

Chapter 5—*Case Studies.*

In Chap. 5, we apply the notions—especially precision complexity—of our framework to several *case studies* in order to shed light upon controversial aspects thereof. Specifically, we consider a ray-tracing model of computation (Sect. 5.2) and a quantum-adiabatic system (Sect. 5.3) that both purport to compute undecidable functions, and the disagreement (Sect. 5.4) over whether restriction to finite precision nullifies the Kochen-Specker theorem; in Sect. 5.5, we list other

such disputes as candidates for future work. Finally, in Sect. 5.6, we abstract some general features from our specific case studies, as well as discussing the extent to which the disputes have in fact been settled.

Chapter 6—*Discussion.*

In Sect. 6.1, we consider not the *direct* use of our framework (as a means of quantifying and comparing systems' complexity), but rather its *indirect* use in definitions of analogues of concepts—cryptography, Kolmogorov complexity and set theory—connected with complexity theory. In Sect. 6.2, we view from the perspective of our framework various fundamental questions regarding the nature of computation—whether complexity is inherent in problems, where the computer/environment boundary lies, whether there exists a single underlying computational resource, etc. We summarize below (in the remainder of Sect. 6.3) some possible directions for future work and add some concluding comments.

6.3.2 Future Work

We list here the topics suggested throughout this dissertation for extension of the present project; given in brackets are references to the topics' discussion above.

- A slime-mould implementation of Shor's factorization algorithm (Footnote 14 of Chap. 2);
- modification of the mass-comparison system of Sect. 3.1.2 to facilitate recognition of equality (Footnote 10 of Chap. 3);
- formulation of unconventional paradigms' power as oracular consultation (Sect. 3.2.1 *Features of Computational Models*);
- use of joint entropy or similar to determine which parameters contribute to precision (Sect. 3.3.3 *Error; Precision Complexity*);
- exploitation of chaos to improve precision (final aside of Sect. 3.3);
- the precision costs incurred in the computational use of quantum walks (Sect. 3.6.2 *Quantum Walks*);
- the existence or otherwise of a fundamental resource of which all others are facets (Sect. 3.8);
- the correspondence between the new and traditional hierarchies of complexity classes, and the 'transfer theorems' that this correspondence implies (Sects. 4.3.5 and 4.3.6);
- Gap-Theorem-like conjectures, the generalization of these conjectures to ordinals, resource-constructibility, and related issues (Sect. 4.4.3);
- a 'dual' of normalization that corrects understatement, rather than exaggeration, of resources' impact (Remark 41), and the conjectured incompleteness of finite sets of such corrections (Footnote 29 of Chap. 4);

- normalization as a topic in traditional complexity theory (final aside of Sect. 4.5);
- continuation of the resolution of the Cabello/Meyer dispute (Sect. 5.4.2);
- formalization (especially of precision aspects) of controversies and other issues concerning [35, 42, 84, 91], etc. (Sect. 5.5);
- further detail concerning 'side-issues' such as cryptography, Kolmogorov complexity and set theory (Sect. 6.1); and
- fundamental questions about the nature of computers and computation: inherence of complexity, computer/environment boundaries, underlying resource, etc. (Sect. 6.2).

Of course, there are many other directions in which this work may be developed that we do not mention above; these include

- polynomially restricting the precision of analogue devices similar to those of Chap. 2 so as to ascertain which conic sections can be implemented;[5]
- investigating the complexity of a single problem (e.g., factorization) as approached using several computational paradigms, with a view to understanding resource trade-offs; and
- exploring the probabilistic aspects of our framework, which necessarily arise since non-determinism is permitted of our computers.

6.3.3 Final Comments

We hope to convey to the reader by what is written above that the profound differences between conventional and unconventional computers warrant for the latter an approach to computational complexity theory distinct from that which is practised for the former. We hope, furthermore, that the above satisfies the reader that our notions—resource (especially precision), dominance, $\mathcal{R}$-complexity, normality, etc.—offer a framework of complexity theory suitable for unconventional computers, though maintaining the conventional as a special case. We hope, also, to demonstrate via the consideration above of case studies that the framework is not only of theoretical interest, but also of practical applicability.

We hope, above all, that the work described in this dissertation is of interest and use to complexity theorists, to practitioners of unconventional computing and to the general reader.

[5] The motivation for this suggestion, for which latter we thank Damien Woods, is that this may shed light on the internal structure of P.

Bibliography

[1] S. Aaronson: *Complexity Zoo* website, available at `http://qwiki.stanford.edu/wiki/Complexity_Zoo`

[2] A. Adamatzky (editor): *Int. J. of Unconventional Computing*. Old City Publishing (2005 onwards)

[3] L. Adleman: *Molecular Computation of Solutions to Combinatorial Problems*. Science 266 (1994)

[4] M. Agrawal, N. Kayal, N. Saxena: *PRIMES is in* P. The Annals of Mathematics, second series 160, no. 2 (2004)

[5] S. Akl: *The Myth of Universal Computation*. Queen's University Kingston, School of Computing Technical Report series 2005-492 (2005)

[6] A. Ambainis, O. Regev: *An Elementary Proof of the Quantum Adiabatic Theorem*. arXiv:quant-ph/0411152v2 (2004)

[7] J. Barrow: *New Theories of Everything*. Oxford University Press (2007)

[8] E. Beggs, J. Costa, B. Loff, J. Tucker: *Computational Complexity with Experiments as Oracles*, and *Computational ... II. Upper Bounds*. Proc. of the Royal Soc. A 464 – 465 (2008 – 9)

[9] E. Beggs, J. Costa, B. Loff, J. Tucker: *On the Complexity of Measurement in Classical Physics*. LNCS 4978 (2008)

[10] E. Beggs, J. Costa, J. Tucker: *Limits to Measurement in Experiments Governed by Algorithms*. Math. Struct. in Comp. Sci. 20 (2010)

[11] E. Beggs, J. Costa, J. Tucker: *Physical Oracles: the Turing Machine and the Wheatstone Bridge*. Studia Logica 95 (2010)

[12] E. Beggs, J. Tucker: *Experimental Computation of Real Numbers by Newtonian Machines*. Proc. of the Royal Soc. A 463 (2007)

[13] J. Bekenstein: *Energy Cost of Information Transfer*. Phys. Rev. Lett. 46 (1981)

[14] J. Bell: *On the Einstein Podolsky Rosen Paradox*. Physics 1, no. 3 (1964)

[15] C. Bennett: *The Thermodynamics of Computation—a Review*. Int. J. of Theor. Phys. 21, no. 12 (1982)

[16] C. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W. Wootters: *Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels*. Phys. Rev. Lett. 70 (1993)

[17] C. Bennett, J. Gill: *Relative to a Random Oracle A,* $\mathsf{P}^A \neq \mathsf{NP}^A \neq \mathsf{co\text{-}NP}^A$ *with Probability* 1. SIAM J. Computing 10, no. 1 (1981)

Many of E. Blakey's publications listed below (specifically, [19,21–29]) are available at `http://users.ox.ac.uk/~quee1871/pubs.html`**.**

[18] E. Blakey: *A Model-Independent Theory of Computational Complexity; Price: from Patience to Precision (and Beyond)*. Unpublished dissertation for Transfer of Status (2008)

[19] E. Blakey: *An Analogue Solution to the Problem of Factorization*. Oxford University Computing Laboratory Research Reports series, RR-07-04 (2007)

[20] E. Blakey: *A New Gap Theorem: the Gap Theorem's Robustness against Dominance*. Journal paper based on conference presentation given at *Science and Philosophy of Unconventional Computing* (in preparation) (2010)

[21] E. Blakey: *Apples & Oranges? Comparing Unconventional Computers*. Int. J. of Computers 4, no. 4, W. Mikhael, I. Sandberg, L. Zadeh, A. Kuri-Morales, et al. (editors) (invited paper) (2010)

[22] E. Blakey: *Apples and Oranges? Comparing Unconventional Computers*. New Aspects of Systems Theory & Scientific Computation, N. Mastorakis, V. Mladenov, Z. Bojkovic (editors) (invited paper) (2010)

[23] E. Blakey: *Beyond Blum: What is a Resource?* Int. J. of Unconventional Computing 6, nos. 3 – 4, A. Adamatzky (editor-in-chief), Old City Publishing (2010)

[24] E. Blakey: *Computational Complexity in Non-Turing Models of Computation; The* What*, the* Why *and the* How. Proc. of *Quantum Physics and Logic/Development of Computational Models 2008*, ENTCS series, M. Mislove (managing editor), B. Coecke, P. Panangaden (guest editors), Elsevier (to appear) (2008)

[25] E. Blakey: *Dominance: Consistently Comparing Computational Complexity*. Oxford University Computing Laboratory Research Reports series, RR-08-09 (2008)

[26] E. Blakey: *Factorizing RSA Keys (an Improved Analogue Solution)*. Proc. in Information and Communications Technology 1, Yasuhiro S., Masami H., Hiroshi U., A. Adamatzky (editors), Springer (2008)

[27] E. Blakey: *Factorizing RSA Keys, an Improved Analogue Solution*. New Generation Computing 27, no. 2, Yasuhiro S., Masami H., Hiroshi U., A. Adamatzky (guest editors), Ohmsha/Springer (2008)

[28] E. Blakey: *On the Computational Complexity of Physical Computing Systems*. Unconventional Computing 2007, A. Adamatzky, L. Bull, B. De Lacy Costello, S. Stepney, C. Teuscher (editors), Luniver Press (2007)

[29] E. Blakey (and co-prepared by patent attorneys): *System and Method for Finding Integer Solutions*. United States patent 7453574 (2008)

Many of E. Blakey's publications listed above (specifically, [19,21–29]) are available at `http://users.ox.ac.uk/~quee1871/pubs.html`.

[30] L. Blum, F. Cucker, M. Shub, S. Smale: *Complexity and Real Computation*. Springer (1997)

[31] M. Blum: *A Machine-Independent Theory of the Complexity of Recursive Functions*. J. of the Assoc. for Computing Machinery 14, no. 2 (1967)

[32] D. Bohm, B. Hiley: *The Undivided Universe*. Routledge (1993)

[33] U. Boker, N. Dershowitz: *Comparing Computational Power*. Logic J. of the IGPL 14, no. 5 (2006)

[34] A. Borodin: *Computational Complexity and the Existence of Complexity Gaps*. J. of the Assoc. for Computing Machinery 19, no. 1 (1972)

[35] O. Bournez, M. Campagnolo: *A Survey on Continuous Time Computations*. New Computational Paradigms, Springer (2008)

[36] O. Bournez, M. Campagnolo, D. Graça, E. Hainry: *The General Purpose Analog Computer and Computable Analysis are Two Equivalent Paradigms of Analog Computation*. Proc. of TAMC, Springer (2006)

[37] D. Bovet, P. Crescenzi: *Introduction to the Theory of Complexity*. Prentice Hall (1993)

[38] V. Brattka, P. Hertling, K. Weihrauch: *A Tutorial on Computable Analysis*. New Computational Paradigms, Springer (2008)

[39] S. Braunstein, A. Pati: *Quantum Information with Continuous Variables*. Springer-Verlag (2003)

[40] M. Braverman, S. Cook: *Computing over the Reals: Foundations for Scientific Computing*. arXiv:cs/0509042v1 [cs.CC] (2005)

[41] R. Brent: *Recent Progress and Prospects for Integer Factorisation Algorithms*. LNCS 1858 (2000)

[42] S. Bringsjord, J. Taylor: $\mathsf{P} = \mathsf{NP}$. arXiv:cs/0406056v1 [cs.CC] (2004)

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

[44] A. Cabello: *Finite-Precision Measurement Does Not Nullify the Kochen-Specker Theorem*. Phys. Rev. A 65 (2002)

[45] C. Calude, L. Staiger: *A Note on Accelerated Turing Machines*. Math. Struct. in Comp. Sci. 21 (2010)

[46] L. Cardelli: *Algebras and Languages for Molecular Programming.* LNCS 6079 (2010)

[47] J. Carlson, A. Jaffe, A. Wiles (editors): *The Millennium Prize Problems.* Amer. Math. Soc. and Clay Math. Inst. (2006)

[48] G. Chaitin: *Meta Maths: The Quest for Omega.* Atlantic Books (2006)

[49] A. Church: *An Unsolvable Problem of Elementary Number Theory.* Amer. J. of Math. 58 (1936)

[50] B. Coecke: *Quantum Computer Science* course, Oxford University Computing Laboratory (2006)

[51] The website for the workshop *Complexity Resources in Physical Computation* is available at `http://www.comlab.ox.ac.uk/CRPC09/`

[52] R. Courant, H. Robbins: *What is Mathematics? An Elementary Approach to Ideas and Methods.* I. Stewart (editor), Oxford University Press (1996)

[53] J. Crutchfield, K. Wiesner: *Intrinsic Quantum Computation.* Phys. Lett. A 374, no. 4 (2008)

[54] S. Das, R. Kobes, G. Kunstatter: *Energy and Efficiency of Adiabatic Quantum Search Algorithms.* J. of Phys. A: Math. Gen. 36 (2003)

[55] M. Davis: *Hilbert's Tenth Problem is Unsolvable.* Amer. Math. Monthly 80 (1973)

[56] L. de Broglie: *Rapport au 5e Conseil de Physique Solvay, Brussels.* Proc. thereof (1927)

[57] D. Deutsch: *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer.* Proc. of the Royal Soc. A 400 (1985)

[58] W. Diffie, M. Hellman: *New Directions in Cryptology.* IEEE Trans. on Information Theory IT-22, no. 6 (1976)

[59] B. Douglas, J. Wang: *Can Quantum Walks Provide Exponential Speedups?* arXiv:0706.0304v1 [quant-ph] (2007)

[60] B. Douglas, J. Wang: *Classically Efficient Graph Isomorphism Algorithm using Quantum Walks.* arXiv:0705.2531v1 [quant-ph] (2007)

[61] A. Einstein, B. Podolsky, N. Rosen: *Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?* Phys. Rev. 47 (1935)

[62] A. Ekert: *Quantum Cryptography Based on Bells Theorem.* Phys. Rev. Lett. 67 (1991)

[63] E. Farhi, J. Goldstone, S. Gutmann: *Quantum Adiabatic Evolution Algorithms with Different Paths.* arXiv:quant-ph/0208135v1 (2002)

[64] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser: *Quantum Computation by Adiabatic Evolution.* arXiv:quant-ph/0001106v1 (2000)

[65] R. Feynman: *Simulating Physics with Computers*. Int. J. of Theor. Phys. 21 (1982)

[66] The website for *FOPARA '09*, the workshop *Foundational and Practical Aspects of Resource Analysis*, is available at `http://www.aha.cs.ru.nl/fopara/`

[67] E. Fredkin, T. Toffoli: *Conservative Logic*. Int. J. of Theor. Phys. 21 (1982)

[68] R. Gandy: *Church's Thesis and Principles for Mechanisms*. The Kleene Symposium, J. Barwise, H. Keisler, K. Kunen (editors), North-Holland Publishing Company (1980)

[69] R. Geroch, J. Hartle: *Computability and Physical Theories*. Foundations of Physics 16, no. 6 (1986)

[70] J. Gorecki, J. Gorecka: *Chemical Programming in Reaction-Diffusion Systems*. Proc. of Unconventional Computing: From Cellular Automata to Wetware (2005)

[71] D. Graça: *Computability Via Analog Circuits*. Proc. of Int. Conf. on Computability and Complexity in Analysis (2003)

[72] T. Haist, W. Osten: *An Optical Solution for the Traveling Salesman Problem*. Optics Express 15, no. 16 (2007)

[73] Hanabusa I.: *Blind Monks Examining an Elephant*. Library of Congress catalogue, control no. 2004666374. Illustration in Itchō Kyōgashū, Niigata-Ken Koshi-Gun Nagaoka: Meguro Jūrō; Tōkyō: Dō Shiten, Meiji 21 (1888)

[74] J. Hartmanis: *On the Weight of Computation*. Bulletin of the EATCS 55 (1995)

[75] B. Hendy: Presentation at the Photo Marketing Assoc./Digital Imaging Marketing Assoc. conf. (1998)

[76] I. Herstein: *Topics in Algebra*. Wiley (1975)

[77] D. Hilbert: *Mathematical Problems*. Bull. Amer. Math. Soc. 8 (1902)

[78] C. Hoare, R. Milner (editors): *Grand Challenges in Computing*. British Computer Soc. (2004)

[79] A. Hodges: *Can Quantum Computing Solve Classically Unsolvable Problems?* arXiv:quant-ph/0512248v1 (2005)

[80] R. Hornbeck: *Numerical Methods*. Prentice-Hall (1982)

[81] Z. Ibrahim, Yusei T., Osamu O., M. Khalid: *Experimental Implementation of Direct-Proportional Length-Based DNA Computing for the Shortest Path Problem*. Unconventional Computing 2005: From Cellular Automata to Wetware, C. Teuscher, A. Adamatzky (editors), Luniver Press (2005)

[82] B. Jacobs: *On Generalized Computational Complexity.* J. of Symbolic Logic 42, no. 1 (1977)

[83] R. Jozsa: discussion with Richard Jozsa, 3.iv.2007 (2007)

[84] I. Kanter: *Undecidability Principle and the Uncertainty Principle Even for Classical Systems.* Phys. Rev. Lett. 64, no. 4 (1990)

[85] V. Kendon: personal correspondence with Viv Kendon, Leeds University, 14.i.2008 (2008)

[86] V. Kendon, K. Nemoto, W. Munro: *Quantum Analogue Computing.* Philosophical Trans. of the Royal Soc. A 368 (2010)

[87] T. Kieu: *Hypercomputation with Quantum Adiabatic Processes.* TCS 317 (2004)

[88] T. Kieu: *Quantum Adiabatic Algorithm for Hilbert's Tenth Problem.* arXiv:quant-ph/0310052v2 (2003)

[89] S. Kochen, E. Specker: *The Problem of Hidden Variables in Quantum Mechanics.* J. of Mathematics and Mechanics 17, no. 1 (1967)

[90] Kojiro K.: *On Proving Time Constructibility of Functions.* Theoretical Computer Science 35 (1985)

[91] G. Kreisel: *A Notion of Mechanistic Theory.* Synthese 29 (1974)

[92] R. Landauer: *Irreversibility and Heat Generation in the Computing Process.* IBM J. of Res. and Dev. 5, no. 3 (1961)

[93] M. Li, P. Vitányi: *Introduction to Kolmogorov Complexity and its Applications.* Springer (1997)

[94] S. Lloyd, H. Pagels: *Complexity as Thermodynamic Depth.* Ann. Phys. 188 (1988)

[95] K. Martin: *Entropy as a Fixed Point.* Theor. Comp. Sci. 350 (2006)

[96] D. Meyer: *Finite-Precision Measurement Nullifies the Kochen-Specker Theorem.* Phys. Rev. Lett. 83 (1999)

[97] W. Miehle: *Link-Length Minimization in Networks.* Operations Research 6, no. 2 (1958)

[98] G. Moore: *Cramming More Components onto Integrated Circuits.* Electronics (1965)

[99] M. Nielsen, I. Chuang: *Quantum Computation and Quantum Information.* Cambridge University Press (2000)

[100] N. Nisan: $\mathsf{RL} \subseteq \mathsf{SC}$. Proc. of ACM Symposium on Theory of Computing (1992)

[101] D. Normann: *Applications of the Kleene-Kreisel Density Theorem to Theoretical Computer Science.* New Computational Paradigms, Springer (2008)

[102] P. Orponen: *A Survey of Continuous-Time Computation Theory*. Advances in Algorithms, Languages, and Complexity, Springer (1997)

[103] C. Papadimitriou: *Computational Complexity*. Addison-Wesley (1995)

[104] G. Păun: *From Cells to (Silicon) Computers, and Back*. New Computational Paradigms, Springer (2008)

[105] R. Penrose: *The Emperor's New Mind*. Oxford Paperbacks (1999)

[106] M. Potter: *Sets: an Introduction*. Oxford University Press (1990)

[107] M. Pour-El: *Abstract Computability and Its Relation to the General Purpose Analog Computer (Some Connections Between Logic, Differential Equations and Analog Computers)*. Trans. of Amer. Math. Soc. 199 (1974)

[108] V. Pratt: *Every Prime has a Succinct Certificate*. SIAM J. on Computing 4 (1975)

[109] R. Raussendorf, D. Browne, H. Briegel: *Measurement-Based Quantum Computation on Cluster States*. Phys. Rev. A 68, no. 2 (2003)

[110] J. Reif, J. Tygar, Akitoshi Y.: *Computability and Complexity of Ray Tracing*. Discrete and Computational Geometry 11, no. 1 (1994)

[111] '*Resource*' dictionary entry. *Oxford English Dictionary*, second edition. Oxford University Press (1989)

[112] R. Rivest, A. Shamir, L. Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM 21, no. 2 (1978)

[113] T. Rudolph: personal correspondence with Terry Rudolph, Imperial College London, 3.iv.2009 (2009)

[114] A. Salamon: personal correspondence with András Salamon, Oxford University Computing Laboratory, 31.x.2008 (2008)

[115] A. Salamon: personal correspondence with András Salamon, Oxford University Computing Laboratory, 1 – 11.ii.2010 (2010)

[116] A. Schönage: *On the Power of Random Access Machines*. LNCS 71 (1979)

[117] C. Shannon: *Mathematical Theory of the Differential Analyzer*. J. Math. Phys. MIT 20 (1941)

[118] P. Shor: *Polynomial Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM J. on Computing 26 (1997)

[119] H. Siegelmann, A. Ben-Hur, S. Fishman: *Computational Complexity for Continuous Time Dynamics*. Phys. Rev. Lett. 83, no. 7 (1999)

[120] M. Sipser: *Introduction to the Theory of Computation*. PWS (1997)

[121] R. Stearns, H. Hunt: *Resource Bounds and Subproblem Independence*. Theory of Comput. Syst. 6, no. 38 (2005)

[122] S. Stepney: *The Neglected Pillar of Material Computation*. Physica D: Nonlinear Phenomena 237 (2008)

[123] W. Stevens: *Logic Circuits in a System of Repelling Particles*. From Utopian to Genuine Unconventional Computers, A. Adamatzky, C. Teuscher (editors), Luniver Press (2006)

[124] Tetsu S., Atsushi T., Toshiyuki N., Yoshiki K.: *Amoebae Anticipate Periodic Events*. Phys. Rev. Lett. 100, no. 1 (2008)

[125] B. Trachtenbrot: *Complexity of Algorithms and Computation*. Novosibirsk (1967)

[126] J. Traub: *On Reality and Models*. Boundaries and Barriers: on the Limits to Scientific Knowledge, J. Casti, A. Karlqvist (editors), Addison-Wesley (1996)

[127] A. Turing: *On Computable Numbers, with an Application to the Entscheidungsproblem*, and *On Computable ... A Correction*. Proc. London Math. Soc. 2, nos. 42 – 43 (1936 – 7)

[128] A. Vergis, K. Steiglitz, B. Dickinson: *The Complexity of Analog Computation*. Mathematics and Computers in Simulation 28, no. 2 (1986)

[129] P. Vitányi: *Time, Space, and Energy in Reversible Computing*. Proc. 2nd Conf. on Computing Frontiers, (2005)

[130] C. Walter: *Kryder's Law*. Scientific Amer. 293, no. 2 (2005)

[131] P. Welch: *Discrete Transfinite Computation Models*. Computability in Context, Imperial College Press (2011)

[132] D. Welsh: *Codes and Cryptography*. Oxford University Press (1988)

[133] D. Woods: *Computational Complexity of an Optical Model of Computation*. PhD thesis, National University of Ireland, Maynooth (2005)

[134] D. Woods: *Optical Computing and Computational Complexity*. Proc. of the Fifth Int. Conf. on Unconventional Computation, LNCS 4135 (2006)

[135] Yao Q.: *Classical Physics and the Church-Turing Thesis*. J. of ACM 50 (2003)

[136] T. Young: *Experiments and Calculations Relative to Physical Optics*. Philosophical Trans. of the Royal Soc. of London 94 (1804)

[137] W. Zurek: *Thermodynamic Cost of Computation, Algorithmic Complexity and the Information Metric*. Nature 341 (1989)