Axiomatising physical experiments as oracles to algorithms

Edwin J. Beggs[a], José Félix Costa[b*], and John V. Tucker[c]

[a]Department of Mathematics, College of Sciences, Swansea University, Swansea SA2 8PP, Wales, United Kingdom

[b] Instituto Superior Técnico, Universidade Técnica de Lisboa and Centro de Matemática e Aplicações Fundamentais do Complexo Interdisciplinar, Universidade de Lisboa

[c]Department of Computer Science, College of Sciences, Swansea University, Swansea SA2 8PP, Wales, United Kingdom

Abstract. Earlier we developed a theory of combining algorithms with physical systems based upon using physical experiments as oracles to algorithms. Although our concepts and methods are general, each physical oracle requires its own analysis based upon some fragment of physical theory that specifies the equipment and its behaviour. For specific examples of physical system (mechanical, optical, electrical) the computational power has been characterised using non-uniform complexity classes. The power of the known examples vary according to assumptions on precision and timing but seem to lead to the same complexity classes, namely $P/\log\star$ and $BPP//\log\star$. In this paper we develop sets of axioms for the interface between physical equipment and algorithms that allow us to prove general characterisations, in terms of $P/\log\star$ and $BPP//\log\star$, for large classes of physical oracles, in a uniform way. Sufficient conditions on physical equipment are given that ensure a physical system satisfies the axioms.

1. Introduction

1.1. Oracles

In computability theory, the basic operations of algorithmic models — Turing machines, register machines, recursion schemes, etc. — may be extended with sets,

*The research of José Félix Costa is supported by Fundação para a Ciência e Tecnologia, Financiamento Base 2010 – ISFL–1–209.

or functions, called "oracles." For example, in Post's standard abstraction of Turing's original conception [21], any set S can be used as an oracle in an algorithm as follows: from time to time, in the course of a computation, an algorithm produces a datum x, asks "Is $x \in S$?" and receives an exact answer "yes" or "no" in a single step of the computation.[2]

Adding an oracle increases the computational power of an algorithm. The idea of coupling an oracle to an algorithm is surprisingly subtle and fruitful mathematically. The idea of Turing reduction led Emil Post to a theory of relative computation and the classification of degrees of algorithmic unsolvability.[3] The theory of Turing degrees of unsolvability, such as those below $0', 0'', \ldots, 0^{\omega}$ map much of what is non-computable and have highly important applications to decision problems in logic, mathematics, and programming.[4]

The notion of oracle has been just as influential in complexity theory. The notions of reduction and degrees led to the use of oracles to create the *polynomial time hierarchy* in 1972.[5] Early on, in 1975, came the Baker-Gill-Solovay Theorems [1] that showed that the $P = NP$ question relativizes both positively and negatively and therefore that any proof technique that is unaffected by the addition of an oracle will fail to answer the $P = NP$ question.[6]

Perhaps what is not clear from the mathematical theory is that adding an oracle to an algorithm is a computational idea that occurs *naturally*. One simple, yet seemingly familiar example, is introducing randomness by tossing a coin at various stages in a computation. Properly described, this is using an experiment as an oracle. In the theory of probabilistic Turing machines and complexity classes, such as BPP,

[2] As Turing suggested in [21] §4, the basic theorems of computability, such as undecidability of termination, can be proved for these S-computable functions. Actually, there is nothing special about the operations chosen as basic in an algorithmic model as far as their elementary theory is concerned — a fact best exploited in computability theories over abstract algebras ([18, 19, 20]).

[3] A set A is *Turing reducible* to set B if one can decide membership of A using an algorithm that allows oracle calls to decide membership in B.

[4] The classification of the non-computable in mathematics, using many types of hierarchies and degrees, is a huge achievement, mathematically and philosophically. For many years, the theory of the degrees was considered to be the mathematical eptiome of computability theory because of the remarkable nature of priority methods — arguably the most complex algorithms known — that were developed to solve Post's Problem.

[5] The polynomial time hierarchy was implicit in Karp [15] and was formulated explicitly for the first time in Meyer and Stockmeyer [16], as a possible way of classifying problems not known to be in NP; see also the initial studies [17, 22].

[6] The theorem is: *There exist computable oracles A and B such that $P(A) = NP(A)$ and $P(B) \neq NP(B)$*. In Balcázar, Días and Gabarró [3] there is an excellent introduction to the subject of positive and negative relativisation results for oracles and complexity theory (in Chapters 7 and 8).

randomness is defined by a mathematical oracle (cf. Section 7.4 of Balcázar, Días and Gabarró [3]).

1.2. Physical oracles

Imagine a computation that *from time to time* requires us to consult some external physical device, say, to read some data, make some measurement, choose randomly, guess information, etc. We can view this external interaction as follows: the algorithm formulates a request, query or command, presents it to the device, waits some time for the device to answer, and resumes the computation. The device is likely to function within error bounds so precision is a factor in computation.

In [5] we introduced the idea of taking a physical experiment that measures a physical quantity as a new form of oracle to a Turing machine. We focussed on modelling the structure and operation of these hybrid computational models and, in particular, the question

What can be computed by algorithms boosted by oracles that are physical systems?

In [5], we chose a particular mechanical experiment first studied in [13], the *scatter machine experiment (SME)*, studied the hybrid systems made by combining deterministic Turing machines with the *SME*, and answered the question using non-uniform complexity theory [5, 7]. In subsequent work, we studied several other examples of experiments as oracles, including the *collider machine experiment (CME)* [9] and the Wheatstone bridge [10].[7]

The computational power of the physical oracles we have studied vary according to assumptions on precision and timing. Furthermore, we have constructed different models of a single physical experiment (actually the original *SME*) and shown that they can have different computational attributes, such as runtimes, and different computational powers [12]. The questions arise:

How varied are the attributes, such as runtimes, of physical experiments?

How varied are the computational powers of algorithms when boosted by physical oracles?

Given the potentially infinite diversity of physical experiments — and the variation in modelling each one of those experiments — one might expect answers to be infinitely diverse. However, our case studies of different mechanical, electrical, optical experiments as oracles lead us to similar runtimes and to similar complexity classes, especially $P/\log\star$ and $BPP//\log\star$, depending upon assumptions about precision of equipment. Indeed, in [9], we made a conjecture about the runtimes of experiments, namely: *normally, runtime is exponential in the accuracy of measurement.* To begin

[7] Further experiments await publication. We have also argued in [6, 11] that some physical models are examples of algorithms with physical oracle calls.

to classify experimentation for computational purposes, taxonomies for equipment and procedures are needed.

In this paper, we address the classification and "stability" of models of computation with physical oracles. We develop three sets of axioms for experimental measurement that allow us (i) to explore variations in equipment and behaviour and their effects on the quantity measured; and (ii) to prove general characterisation theorems of the power of experiments as oracles. The axioms specify the *interface* between physical equipment and algorithms. The theorems establish the scope and limits of large classes of physical oracles, in a uniform way, in terms of $P/\log\star$ and $BPP//\log\star$. We also give sufficient conditions on physical equipment that ensure a physical system satisfies the axioms (Theorem 3.2).

1.3. Results

The physical oracles are experiments that measure physical quantities represented by real numbers $x \in (0,1)$. The axioms define those features of the experiment and its behaviour that the algorithms may use and depend upon. The axioms for the interface focus on two properties: time taken to obtain a measurement and the precision of the measurement.

First, we consider physical experiments with deterministic behaviour.

Let $\mathcal{G}$ be a class of functions of the form $g : \mathbb{N} \to \mathbb{N}$ which we will use to bound the runtime of the experiment as a function of its precision. We give axioms for any physical experiment that measures $x \in (0,1)$, which defines a class $E(\mathcal{G})$ of physical oracles. We have found the most important class $\mathcal{G}$ of bounds to be that of all exponentials of linear functions,

$$\text{ExpLin} = \{g : \mathbb{N} \to \mathbb{N} | 2^{an+b} \leq g(n) \leq 2^{cn+d}\},$$

(i.e., $g(n) \in 2^{\Theta(n)}$). Hence the most important class of oracles is $E(\text{ExpLin})$.

The first main theorem is this (Theorem 4.7 below).

Theorem A. *Given a decision problem A in $P/\log\star$, there is a real value $x \in (0,1)$ so that any physical oracle in class $E(\text{ExpLin})$ measuring x can be used to decide A in polynomial time. Conversely, any decision problem A that can be solved in polynomial time by a Turing machine with an oracle in class $E(\text{ExpLin})$ is also in $P/\log\star$.*

Next, we consider non-deterministic behaviour of the physical experiment, which can arise for all sorts of reasons, most simply precision. Our axioms involve probability distributions and lead to a new class $Iid(\delta)$ of physical oracles, where Iid stands for *independent identically distributed*. The second main theorem is this (Theorem 5.3 below).

Theorem B. *Given a decision problem A in $BPP//\log\star$, there is a real value $x \in (0,1)$ so that any oracle in class $Iid(\delta)$ measuring $x \in (\delta, 1-\delta)$ can be used by a Turing machine to decide A in polynomial time. Conversely, any decision problem that can be solved in polynomial time by a Turing machine with an oracle in class $Iid(\delta)$ is in $BPP//\log\star$.*

The structure of the paper is this. In Section 2 we describe carefully the characteristics of experiments and our line of thought summarised above. In Section 3 we give the first set of axioms and prove a sufficiency condition for experiments to satisfy the axioms. In Section 4 we prove the lower bound for the first theorem and on altering an axiom on repeatability we prove the upper bound completing the proof of Theorem A. In Section 5 we give a much simplified set of axioms with a probability distribution for the repeatability and prove Theorem B. In Section 6 we reflect on the results.

This study is part of a our programme on the physical foundations of computation, primarily it concerns [5, 7, 9, 10, 12]. Some knowledge of non-uniform complexity classes is needed.

2. Experiments and interfaces

2.1. Algorithms and experiments

The basic structure of an algorithm combined with an experiment as oracle is depicted in Figure 1. The experiment consists of various components:

physical equipment + data + interface + experimental procedure.

The equipment is modelled, based upon some fragment of the physical theory T. The algorithm is modelled by a Turing machine, which is suited to complexity considerations. In this context, here is a working definition of physical oracles.

Definition 2.1. *A physical oracle is a physical device or system with an interface that accepts queries and returns an answer. For an oracle to a Turing machine, the query would be written on a query tape. A time limit may be assigned to such a query, after which the result 'time out' is returned if no other result is obtained in this time.*

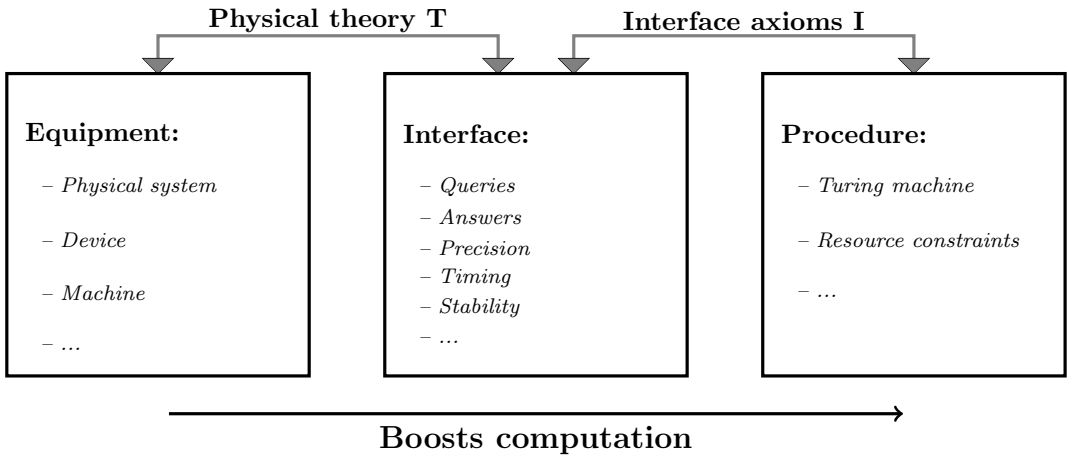

Figure 1. An experiment as oracle.

Now, in developing a theory of algorithms and experiments we note two methodological principles. The equipment involved in making any measurement is potentially diverse. It is made by different manufacturers and according to somewhat abstract specifications of its type and characteristics.

Variation of Equipment *Given any experiment to measure a quantity, its experimental procedure can be applied to a variety of forms of physical equipment to obtain the same measurements with few possible changes to performance.*

This complementary idea is very important in theoretical practice:

Invariance of Models of Equipment *Given any experiment, there is a portfolio of theoretical models for the equipment and its behaviour, employing different physical assumptions, that can be used with the experimental procedure to reason about the experiment.*

Indeed we do not need to know the complete description of the equipment but only what properties are needed to reason correctly about the experimental procedure. We need a specification of the equipment that characterises the interface.

A strong demonstration of this principle can be found in the different models of the wedge in the SME; in particular, in [12] a simple experimental procedure is applied to two distinct models of the simple wedge having dramatically different computational

properties.

2.2. Experiments that measure: six characteristics

Here are the properties that seem to be essential characteristics of the experiments we have used as oracles so far:

1. *Real values.* A physical parameter represented by a real number is being measured — position, inertial mass, location, resistance, mass of an electric particle, temperature, optical angles, etc.

2. *Queries.* Each query expresses a rational, even dyadic, number that is a possible approximate value of the parameter being measured. This test value is then compared to the actual 'physical' value in an experiment. The experiment ultimately implements a decision procedure, such as $<$ or $>$, on physical parameters, possibly leaving equality undecided.

3. *Finite output.* Only qualitative output is allowed in a single experiment: *right* or *left*, *yes* or *no*, *north* or *south*, etc.; or a trivalent output, adding the response *time out*.

4. *Protocol timer.* The protocol between the Turing machine and the experimental apparatus measures the time needed for the experiment to deliver an answer in reasonable cases. It does not guarantee that the experiment will return an answer within the protocol time, and in that case *time out* is returned.

5. *Sufficiency of the protocol.* Enough experiments are completed within the protocol timer to give results that can be used to give more physical information.

6. *Repeatability.* The extent to which identical queries give the same results.

We must also address the problem of the precision of the experiment. For simplicity of explanation, we localise the imprecision in the experiment. Suppose that an experiment to measure a mass x compares it with a test mass y on a balance, and outputs $x > y$ or $x < y$, the value $x = y$ leading to the experiment running out of time. The value of y is specified on the query tape, the experiment is performed, and the result returned to the Turing machine. However, we need to consider the *error in making* a test mass from the instructions on the query tape, and look at the three most obvious possibilities.

1. *No error.* The test value y is exactly the value given on the query tape. More generally, the experiment is carried out exactly as specified on the query tape.

2. *Arbitrary precision.* The value on the query tape comes with a specified error, and the test value y (or the experiment in general) is set up to within that error margin.

3. *Fixed precision.* There is an error margin which it is impossible to improve on – no matter how hard we try y cannot be prepared more accurately than in a certain interval around the value specified on the query tape.

The no error case is an interesting idealisation, but is of little practical value. However, the arbitrary precision case has, in many experiments, identical computational power. To practically implement arbitrary precision would, at the very least, take more time for a smaller error margin, but accounting for this additional time is part of the function of the protocol.

The fixed precision case seems doomed from the start – surely there is only a finite amount of information that could be extracted from such an experiment and so could be appended as part of the program. However, we now turn to probability: we repeat the experiment many times and extract statistical data. To do this requires (more or less) assumptions about independence and identical distribution of repeats of the experiment. If this sounds unlikely in practice, remember that we, according to current understanding of quantum theory, do have a source of 'real randomness' which we can use to 'kick' the apparatus between experiments. Indeed, in practice, the quantum nature of the world we live in is quite likely to randomise the experimental setup to some extent, whether we want this or not.

It is a striking fact that in many experiments, this fixed precision statistical approach can have identical computational power to the no error case. Probabilistic Turing machines clocked in polynomial time give BPP as a probabilistic complexity class. If we add a physical oracle of type $E(\mathrm{ExpLin})$ we then get $BPP//\log\star$. But, by Theorem B, this is the same as we get from the oracle $Iid(\delta)$.

2.3. The interface and its protocol

In contrast to the complexity of the equipment inside the physical oracle, the *interface protocol*, which is all the Turing machine sees, could hardly be simpler. There is a query tape, and when the Turing machine enters the query state, it rests while the experiment proceeds. When the result is returned the Turing machine resumes its calculation. In fact, there are only three differences between the protocols for standard oracles and physical oracles as far as the Turing machine is concerned.

1. *Runtimes.* The time taken by the physical oracle to reply is rarely one or a constant number of cycles; it varies. It will be convenient for us to assume that the time taken is a function of the length of the query. Thus a polynomial function of the

length gives a *polynomial protocol*, and an exponential function of the length gives an *exponential protocol*. For simplicity the result is only returned at the end of the time limit given by the protocol, even if the experiment was finished much more quickly.

2. *Timeouts.* To keep things simple, we consider only two outcomes for an experiment: *yes* and *no*. However, an experiment takes time to perform, which may depend on the detail of the result being sought or the experiment may not be completed in any finite time (e.g., comparing two masses on a balance when the masses are actually equal). To stop the Turing machine waiting indefinitely for a result that may never come, we impose time limits on the experiments for which we need another outcome *time out* for an experiment that was terminated before an answer was obtained.

3. *Nondeterminism.* In the probabilistic case, identical queries may produce different results. This is not new, as coin tossing oracles have been used for a long time.

2.4. Timing

As experimental scientists will only too readily testify, experiments and observations are prone to errors and entail costs. We focus on only one of these costs, the time taken. In previous papers analysing physical experiments, we were led to a conjecture relating the accuracy of a measurement to the time taken to perform the measurement. These experiments had the form of comparing a test value y to the real world value x of a physical quantity, this comparison taking experimental time

$$\frac{A}{|x-y|^q} \;\leq\; T_{\text{experiment}}(x,y) \;\leq\; \frac{B}{|x-y|^n}\ . \tag{1}$$

Here $n, q \geq 1$ and $A, B > 0$ are constants determined by the experiment. Now the two inequalities in (1) have a different status. They were arrived at by considering models of experiments that omitted many unrealistic features (such as infinitely sharp points), but were still idealised. The lower bound for the experimental time can be considered much more definite – it is extremely difficult to see how to get around such a bound, even allowing for idealised thought experiments. The upper bound is, in reality as opposed to thought experiments, much more questionable, but is a reasonable working hypothesis.

Some experiments can take a long time or even fail to return a result. The time that the machine is allowed to wait for the results of the experimental procedure should not be unlimited. We wish to place bounds on execution times of Turing machines. To control the physical oracle, the protocol will be equipped with a special *protocol clock*, based upon a time constructible function. A function $T : \mathbb{N} \to \mathbb{N}$ is said to be

time constructible if there exists a Turing machine that, for all inputs of size n, halts exactly in $T(n)$ steps. Let $\mathcal{M}$ be a Turing machine, equipped with oracle tape and oracle $\mathcal{O}$. Let $\mathcal{T}$ be the Turing machine which witnesses the runtime of computations as a function of the size of the input w, i.e., a system clock, and let $\mathcal{P}$ be the Turing machine that witnesses the protocol time as a function of the size of the query z, i.e., a query timer. A Turing machine, with an internal system clock $\mathcal{T}$ and a timing clock $\mathcal{P}$ for running the experiment, which controls the apparatus and returns results may then be specified. Assuming that all the transitions of the composite Turing machine can be made to happen in the physical time unit, we have the connection between machine time and physical time.

However, we should be aware that such a composite Turing machine does not measure time! Indeed, an experiment to measure time with n bits of precision (e.g., the period of an orbital motion) takes time at least exponential on n, in agreement with our theory. The former machine time is a *more or less time*, an order of magnitude of time. And since our experiments can be done in time of exponential order, some exponential of the class will do!

Our experiments are fundamental measurements, centred only on observable events occurring in the physical apparatus; they do not depend on precise measurements of time, nor are they a result of operations between values of measurables. In fundamental measurement, the value is not obtained as a result of (say, arithmetical) operations on other measurements (e.g., using time to get more accuracy). These measurements are said to be derived, because their procedures rest on a priori theories relating the measurables.

The reader may wonder where formula (1) comes from. To provide a general idea on how time varies with the required precision, just consider a balance with two pans for the purpose of measuring mass, and a toolbox of standards m_2 designed to measure mass m_1 with some precision. The relevant events referred above are, in this case, "right/left pan up/down". To detect an event, the experimenter has to wait some time, for the acceleration of a pan is proportional to $|m_1 - m_2|$ and the time needed to detect a downward movement of, let us say, 1 mm is proportional to $1/|m_1 - m_2|^{1/2}$. Thus, if the difference of the masses in binary is 2^{-an}, for some constant a, for a precision of n bits, the time ellapsed is proportional to 2^{bn}, for some constant b, that is exponential on the number of bits of precision. This calculation gives a lower bound on the time. An upper bound can then be fixed for values of the mass that, in this paper, codes for advice functions. Other experiments, being more or less sophisticated, reveal the same timing law (1). Thus, it became a conjecture that we discuss in detail in [8].

3. Implementing the axioms for $E(\mathcal{G})$.

Here we take the description in Section 2.2, and interpret it in detail for a particular case, called $E(\mathcal{G})$. Here $\mathcal{G}$ is a class of functions, where each $g \in \mathcal{G}$ is a function $g : \mathbb{N} \to \mathbb{N}$.

Definition 3.1. *The following axioms define the class of physical oracles $E(\mathcal{G})$.*

1. Real values. *The experiment is designed to find a physical parameter $x \in (0, 1)$.*
2. Queries. *Each query string is interpreted as a binary string of 0s and 1s, $y_1y_2 \dots y_k$, giving a dyadic rational $y = 0{\cdot}y_1y_2 \dots y_k$.*
3. Finite output. *The result is either $y < x$, $y > x$ (correctly assigned) or time out.*
4. Protocol timer. *There is a $g \in \mathcal{G}$ so that the time taken for any query of length k is bounded by $g(k)$.*
5. Sufficiency of the protocol. *If $|x - y| > 2^{-k}$ for the dyadic rational $y = 0{\cdot}y_1y_2 \dots y_k$, then the result is not time out.*

These axioms for $E(\mathcal{G})$ are sufficient to prove various complexity results for a Turing machine using $E(\mathcal{G})$ as a physical oracle, and we shall do this in Section 4. But are there really physical experiments implementing this specification? The answer is yes: first, we shall describe the common properties of the experiments which show that they implement the axioms, and then give a list of some of the experiments.

The experiments are designed to measure a physical parameter x, which for convenience we shall take to be in the interval $(0, 1)$. A test value y is compared to the value of x in the experiment, and we get the results $y < x$ or $y > x$. The time taken for this experiment is just that given in (1), which we repeat:

$$\frac{A}{|x-y|^q} \;\leq\; T_{\text{experiment}}(x, y) \;\leq\; \frac{B}{|x-y|^n}\,, \tag{2}$$

for some rationals $A, B > 0$ and integers $n, q \geq 1$. Now you can see why there is no $x = y$ result, as if we put $x = y$ in the formula for the time, the result is infinite.

But now we must consider the whole system behind the physical oracle. The experimenters receive a message from the protocol, and must set up the experiment, perform the experiment, and report the answer back to the protocol. The experimenters have to set up the test value $y = 0{\cdot}y_1y_2 \dots y_k$ for the experiment, and this

will take time. It may also entail error, that is the actual value of the test variable in the experiment y' may not be exactly the requested value $y = 0 \cdot y_1 y_2 \ldots y_k$. We shall suppose that we either have no error, or arbitrarily small error, i.e., that the experiment can be set up to any desired finite accuracy.

But can the oracle return the wrong answer: because of the error in setting up the test value, the reported answer might be $y > x$ whereas in reality $y < x$? We can eliminate this possibility by making the experimental time work in our favour, by ensuring that all experiments which might report incorrect results instead report *time out*. If we have such a possible incorrectness, then x must lie between y and y', so

$$|y - y'| \geq |x - y'| \ .$$

Now, by (2),

$$T_{\text{experiment}}(x, y') \ \geq \ \frac{A}{|x - y'|^q} \ \geq \ \frac{A}{|y - y'|^q} \ .$$

If the experimenters impose a time limit on the experiment, then an incorrect case must *time out* if

$$\text{experiment time limit} \ < \ \frac{A}{|y - y'|^q} \ . \tag{3}$$

Now we address the sufficiency of the protocol. If $|x - y| > 2^{-k}$, and supposing that $|y - y'| \leq 2^{-k-1}$, we have $|x - y| > 2^{-k-1}$. Now, by (2),

$$T_{\text{experiment}}(x, y') \ \leq \ \frac{B}{|x - y'|^n} \ < \ 2^{n(k+1)}\, B \ .$$

Thus, the experiment cannot *time out* if

$$\text{experiment time limit} \ \geq \ 2^{n(k+1)}\, B \ . \tag{4}$$

We set the time limit on the experiment to be $2^{n(k+1)}\, B$, and in that case we require $|y - y'| \leq 2^{-k-1}$ and (3) for correctness, and that now gives

$$|y - y'| \ < \ \min\left\{ 2^{-n(k+1)/q}\ (A/B)^{1/q}\, ,\ 2^{-(k+1)} \right\} \ .$$

It will be convenient (by taking more accuracy than we actually need) to give a fixed integer $r \geq 1$(determined by A, B, n, q) so that $2^{-r(k+1)}$ is less than the above minimum. Now, if we specify accuracy $|y - y'| < 2^{-r(k+1)}$ then we get correctness of the answer and sufficiency for the experiment.

But how long does this whole process take? We have a time limit for the experiment, but how long does it take to set up the experiment? We assume that the time taken to set up the experiment with the actual test value y' to within an error $|y - y'| < 2^{-s}$ of the query value $y = 0{\cdot}y_1y_2 \ldots y_k$ is

$$T_{\text{setup}}(y, s) \;\leq\; 2^{p\,\max\{s,k\}}\, E \;, \tag{5}$$

for some rational $E > 0$ and integer $p \geq 1$. Thus we can set up the experiment for our specified accuracy $|y - y'| < 2^{-r(k+1)}$ in time

$$\text{set up time limit} \;=\; 2^{pr(k+1)}\; E \;.$$

The total time for the whole procedure, the value which we can give to the protocol timer, is

$$g(k) \;=\; \text{set up time limit} + \text{experiment time limit} \;=\; 2^{pr(k+1)}\; E + 2^{n(k+1)}\, B \;. \tag{6}$$

Alternatively, we can take a single exponential $2^{u(k+1)}$ bounding this whole expression. We have then proved the following theorem:

Theorem 3.2 (**Interface Verification Test 1**)**.** *Suppose we are given an experimental procedure to measure $x \in (0, 1)$ in the following manner. Suppose there are rationals $A, B, E > 0$ and integers $n, q, p \geq 1$ so that*

(a) Given $y = 0{\cdot}y_1y_2 \ldots y_k$ we can set up the experiment with a test value $y' \in \mathbb{R}$ with $|y - y'| < 2^{-s}$ in time $2^{p\,\max\{s,k\}}\, E$.

(b) We can run the experiment to determine if $x < y'$ or $x > y'$ in time

$$\frac{A}{|x - y'|^q} \;\leq\; T_{\text{experiment}}(x, y') \;\leq\; \frac{B}{|x - y'|^n} \;.$$

Then there is a physical oracle to a Turing machine using the experiment which obeys the axioms for $E(\text{ExpLin})$.

Here is a list of some published experiments which obey the experimental time constraint in Theorem 3.2.

1. *The collider machine.* This is designed to measure the inertial mass x of a particle in classical mechanics. A particle of mass y is prepared and collided with the original particle [9].

2. *The Wheatstone bridge.* This is designed to measure an electrical resistance x. A resistance y is prepared and compared with the original resistance [10].

3. *The general scatter machine.* This is designed to measure a position x on an interval, by comparing it to a dyadic position y [12].

4. The computational power of E(ExpLin)

4.1. The non-uniform complexity class P/log⋆

These complexity classes are based on *advice functions*. We assume that there is a single algorithm for inputs of every size, which is aided by certain information, called *advice*, which may vary for inputs of different sizes. The advice is obtained via a function $f : \mathbb{N} \to \Sigma^*$. For each input w, it is combined $\langle w, f(|w|)\rangle$ by means of a pairing function $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$. This function and its inverses $(\cdot)_1$ and $(\cdot)_2$ are computable in linear time.[8]

Definition 4.1. *Let $\mathcal{B}$ be a class of sets and $\mathcal{F}$ a class of functions. The advice class $\mathcal{B}/\mathcal{F}$ is the class of sets A for which there exists $B \in \mathcal{B}$ and some $f \in \mathcal{F}$ such that, for every $w \in \Sigma^*$, $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$.*

The set $\mathcal{F}$ is called the *advice class* and f is called the *advice function*. Examples for $\mathcal{B}$ are P and BPP.

We will consider two instances for the class $\mathcal{F}$: *poly* is the class of functions with polynomial size values, i.e., *poly* is the class of functions $f : \mathbb{N} \to \Sigma^*$ such that, for some polynomial p, $|f(n)| \in O(p(n))$; log is the class of functions $g : \mathbb{N} \to \Sigma^*$ such that $|g(n)| \in O(\log(n))$. We will also need prefix non-uniform complexity classes. For these classes we may only use prefix functions, i.e., functions f such that $f(n)$ is always a prefix of $f(n+1)$. The idea behind prefix non-uniform complexity classes is that the advice given for inputs of size n must also be useful to decide smaller inputs.

Definition 4.2. *Let $\mathcal{B}$ be a class of sets and $\mathcal{F}$ a class of functions. The prefix advice class $\mathcal{B}/\mathcal{F}*$ is the class of sets A for which there exists $B \in \mathcal{B}$ and some prefix function $f \in \mathcal{F}$ such that, for all words $w \in \Sigma^*$ of length less or equal to n, $w \in A$ if and only if $\langle w, f(n)\rangle \in B$.*

As an example of Definition 4.2 we have $P/\log\star$. It is important to notice that the usual non-uniform complexity classes contain undecidable sets, e.g., $P/poly$ contains the sparse halting set. Classes with larger advice usually become uninteresting: exponential advice, for instance, allows a Turing machine to decide any set in linear time. The non-computability of the non-uniform complexity classes results exclusively from the non-computability of the advice functions.

4.2. Coding a log⋆ advice function as a real

Suppose that the real number $1 > x > 0$ has a binary expansion consisting only of the triples 100, 010 or 001. For example, we could have (separating the triples for

[8] By definition, $(\langle w, v\rangle)_1 = w$, $(\langle w, v\rangle)_2 = v$, and $\langle (w)_1, (w)_2\rangle = w$.

clarity)

$$x \;=\; 0{\cdot}100\ 100\ 010\ 001\ 010\ 100\ \ldots$$

Such numbers occur in a Cantor subset of $[0,1]$, and have the useful property that for every such x,

$$|x - k/2^n| \;>\; 1/2^{n+5} \quad \forall k \in \mathbb{Z}\ , \quad \forall n \in \mathbb{N}\ . \tag{7}$$

Now we suppose that we are given an advice function $f : \mathbb{N} \to \Sigma^*$, where Σ is a finite alphabet. We code every letter in Σ as a binary number (for example, Unicode or ASCII). Then replace every 0 in its binary form by 100 and every 1 by 010. Thus a letter in Σ might have binary representation 00101, and on replacement by triples (separating triples for clarity) we would have

100 100 010 100 010 .

If we perform this binary and then triple replacement letter by letter, we get a function $c : \Sigma^* \to \{100, 010\}^*$, where the length of $c(v)$ is bounded by a linear function of the length of $v \in \Sigma^*$.

Now suppose that $f : \mathbb{N} \to \Sigma^*$ is such that each $f(n)$ is a prefix of $f(n+1)$, i.e., $f(n+1) = f(n)\, s$, for some $s \in \Sigma^*$. We code f as a real number $x(f)$ in the Cantor set described above. We recursively define $x(f)$ as the limit of $x(f)(n)$, using the coding $c : \Sigma^* \to \{100, 010\}^*$ by starting with

$$x(f)(0) \;=\; 0 \cdot c(f(0))$$

and using the following cases, where $f(n+1) = f(n)s$,

$$x(f)(n+1) \;=\; \begin{cases} x(f)(n)\, c(s) & \text{if } n+1 \text{ is not a power of 2} \\ x(f)(n)\, c(s)\, 001 & \text{if } n+1 \text{ is a power of 2} \end{cases}$$

The net effect is to add a 001 at the end of the codes for $n = 1, 2, 4, 8 \ldots$. We can now prove the following result:

Proposition 4.3. *Given an advice function f for a decision process in $P/\log\star$, there is a real number $1 > x > 0$ obeying (7) so that, to reconstruct advice sufficient to decide a word w, it is sufficient to read the first logarithmically (in the length of w) many binary digits of x.*

Proof: Construct the real $x(f)$ as above, and note that it obeys (7) by construction. Now, for the word w, take $m \in \mathbb{N}$ with $2^{m-1} < |w| \leq 2^m$. We read the binary

expansion until we have counted a total of $m+1$ 001 triples. Now the extra 001s are deleted, and we can reconstruct $f(2^m)$, which can be used as advice for w. But how many binary digits from $x(f)$ must we read? We start with $|c(f(n))| \leq L\ \log_2(n)+K$ (for some constants K and L) from the definition of log length. This means that we must read at most $L\,m + K + 3\,m$ digits when we add in the 001 separators, so the number of digits is logarithmic in $|w|$. $\square$

4.3. Implementing binary search using an E(ExpLin) oracle

Suppose that we have a decision problem in $P/\log\star$, with advice function f. By Proposition 4.3 there is an $1 > x > 0$ so that to decide a word w we need to read at most $m = K + L\ \log_2(|w|)$ binary places of x. Here $K, L > 0$ are constants and $|w|$ is the length of w. We are given an oracle O in the class $E(\text{ExpLin})$ for the real value $x \in (0,1)$, using a protocol timer $g \in \text{ExpLin}$.

The Turing machine will begin its operation by running a sub-program to find the first $K + L\ \log_2(|w|)$ binary places of x by binary search. Then this is used as advice to run the program to decide the input word w, in polynomial time in $|w|$. All we have to do is to show that we can find the first $m = K + L\ \log_2(|w|)$ binary places of x in polynomial time, using calls of the given oracle O. We suppose that for O's protocol timer g, $g(n) \leq A\,2^{Bn}$.

The principle of binary search is simple: Begin with $y_0 = 0$ and $y_1 = 1$. Call the oracle O with query $y = 1/2$ to see if $x < 1/2$ (in which case take the interval $[0, 1/2]$) or $x > 1/2$ (in which case take the interval $[1/2, 1]$). Now continue bisecting the intervals until we find k with $x \in [k/2^m, (k+1)/2^m]$. The complications with our oracle O are that it may report a *time out* result, and that we have to estimate how much time it takes to answer the queries.

In principle, we need O to answer m queries, of potential length 1 to m. For example, suppose at the third stage of the bisection we need to find if $x > 0{\cdot}010$ or $x < 0{\cdot}010$. If we ask this directly, we might get a *time out* result. To avoid this, add five zeros, and make the query $y = 0{\cdot}01000000$, i.e., $y_1 \ldots y_8 = 01000000$. This query takes not more than time $A\,2^{8\,B}$ to answer. By the *sufficiency of the protocol* property, if $|x - y| > 1/2^8$, we get the correct answer, either $x > 0{\cdot}010$ or $x < 0{\cdot}010$. But (7) states that $|x - y| > 1/2^8$, so we do get the correct answer.

To get the required advice by binary search to find the first m digits, it suffices to have m queries, each taking not more than time $A\,2^{(m+5)\,B}$. This proves the following proposition:

Proposition 4.4 (**Lower Bound**). *Given a decision problem in $P/\log\star$, there is an $x \in (0,1)$ so that any oracle in class $E(\text{ExpLin})$ can be used to solve the problem in polynomial time.*

4.4. An upper bound on the computational power of E(ExpLin)

We have seen that every problem in $P/\log\star$ can be solved by some oracle in the class $E(\text{ExpLin})$ in polynomial time. But is every decision problem solved by a Turing machine augmented by an oracle in class $E(\text{ExpLin})$ necessarily in $P/\log\star$? The answer is no, as is discussed in some detail at the beginning of Section 5, the basic reason being the lack of repeatability, and that this could be used to input more information than expected to the Turing machine. To get round this problem, we consider a second class of oracles:

Definition 4.5. *The following axioms define the class of physical oracles* $E_{\text{det}}(\mathcal{G})$.

1. Real values. *The experiment is designed to find a physical parameter* $x \in (0,1)$.
2. Queries. *Each query string is interpreted as a binary string of 0s and 1s,* $y_1y_2\ldots y_k$, *giving a dyadic rational* $y = 0{\cdot}y_1y_2\ldots y_k$.
3. Finite output. *The result is either* $y < x$, $y > x$ *(correctly assigned) or time out.*
4. Protocol timer. *There is a* $g \in \mathcal{G}$ *so that the time taken for any query of length* k *is bounded by* $g(k)$.
5. Sufficiency of the protocol. *If* $|x-y| > 2^{-k}$ *for the dyadic rational* $y = 0{\cdot}y_1y_2\ldots y_k$, *then the result is not time out.*
6. Repeatability. *If the same query is used on the oracle, the same output will be obtained.*

We now have a slightly modified version of Theorem 3.2, to show that some experiments can give rise to oracles of this class. The proof is just adding repeatability to the list of axioms.

Theorem 4.6 (**Interface Verification Test 2**). *Suppose we are given an experimental procedure to measure* $x \in (0,1)$ *in the following manner. Suppose there are rationals* $A, B, E > 0$ *and integers* $n, q, p \geq 1$ *so that*

(a) Given $y = 0{\cdot}y_1y_2\ldots y_k$ *we can set up the experiment with a test value* $y' \in \mathbb{R}$ *with* $|y-y'| < 2^{-s}$ *in time* $2^{p\max\{s,k\}}E$. *Further, every run of the experiment for a given* y *gives an identical* $y' \in \mathbb{R}$.

(b) We can run the experiment to determine if $x < y'$ *or* $x > y'$ *in time*

$$\frac{A}{|x-y'|^q} \;\leq\; T_{\text{experiment}}(x,y') \;\leq\; \frac{B}{|x-y'|^n}\ .$$

Then there is a physical oracle using the experiment which obeys the axioms for $E(\mathrm{ExpLin})$.

We can now find an upper bound to the computational power of a Turing machine with an oracle of class $E(\mathrm{ExpLin})$. As now we know that the result of a given query is always the same, all we have to do is to make an advice function which encodes the results of all possible queries up to a given query length, and check the length of the advice function. This is a consequence of the *sufficiency of the protocol* and the *correctness* of the answers $x < y$ and $x > y$, which give, for a query of length n,

$$\begin{aligned} y > x + 2^{-n} &\implies \text{result is } y > x \ , \\ y < x - 2^{-n} &\implies \text{result is } y < x \ . \end{aligned} \tag{8}$$

The key is in looking at the results on successive $y = k/2^n$ for integer k and fixed n. For convenience we extend this from $y \in (0, 1)$ to all integer k by filling in the obvious answers. Now we get results $y < x$, *time out* and $y > x$, which we shorten to $<$, 0 and $>$ respectively. Now for sufficiently small k, by (8), we get the result $<$, and sufficiently large k, we have $>$. From (8) the transition zone cannot be longer than three units long, and so can only have a few forms, which we list:

$$\begin{array}{llllll} \ldots & < & > & \ldots & & \\ \ldots & < & 0 & > & \ldots & \\ \ldots & < & * & * & > & \ldots \\ \ldots & < & * & * & * & > \quad \ldots \end{array} \tag{9}$$

Here the $*$ represent some choices of the three symbols (with the first not being a $<$, and the last not being a $>$). To each n we give a finite string (length independent of n) to describe which pattern the transition zone has.

Now, for our given n, we would like to describe where the transition zone starts (say the k giving the $<$ after the dots). By (8) this value of k, which we call k_n, obeys the inequality

$$x - \frac{3}{2^n} \ \leq \ \frac{k_n}{2^n} \ \leq \ x + \frac{1}{2^n} \ . \tag{10}$$

The problem is that an absolute description of k_n cannot be done in a data string which has length independent of n. However from (10) we can deduce that $2\,k_n - k_{n+1}$ is an integer between -7 and 5, so we can simply record this number instead of k_n.

Now, beginning from the $n = 1$ case, we form a word $u(n)$ recursively. The word $u(1)$ describes the results from all queries of length one. Recursively we form $u(n+1)$ by appending to $u(n)$ a word describing the form of the transition for $n + 1$ and the

integer $2\,k_n - k_{n+1}$. As this is a bounded amount of data (independently of n), the length of $u(n)$ is bounded by a constant times n, and it contains enough information to reconstruct the result of any query of length $\leq n$.

Now consider a program using an oracle in the class $E(\mathrm{ExpLin})$ to solve a decision problem in polynomial time in the length of the input. So for input w, the time will be $\leq p(|w|)$, so we can only perform queries of length bounded by a linear function in $\log_2(|w|)$. By the previous paragraph we can use $u(n)$ as an advice function, where n is bounded by the linear function in $\log_2(|w|)$. We have now proved the second half of the following result, the first half being a corollary of Proposition 4.4:

Theorem 4.7. *Given a word decision problem in $P/\log\star$, there is an $1 > x > 0$ so that any oracle in class $E(\mathrm{ExpLin})$ for real value $x \in (0,1)$ can be used to solve the problem in polynomial time. Conversely, any decision problem that can be solved in polynomial time by a Turing machine with an oracle in class $E(\mathrm{ExpLin})$ is in $P/\log\star$.*

5. Removing the repeatability assumption

If we allow the same query to give different results on different occasions, we have to be rather careful about the computational power.

As an extreme case, our innocent looking apparatus might be a radio receiver, tuned to a signal transmitted by little green men from Alpha Centauri. Further, these little green men might have invented a hyper computer, and be transmitting its non-Turing computable output to our low tech receiver at a constant bit rate.

To exclude weird scenarios, and actually get some results, we need some assumptions on the 'non-repeatability' or 'non-determinism' of queries. The easiest assumption is that the non-repeatability is of the form of statistically independent results. That is, if we repeat an identical experiment n times, the probability of the outcome of the $n+1$-th time is completely independent of the previous n results, or indeed the fact that there were any previous experiments at all. So we could return to the axioms of $E(\mathcal{G})$ given in Section 3, remove the repeatability assumption, and continue. However, this is too complicated for our purposes and we only need a much simpler set of axioms. If we have to remove repeatability from what would otherwise be an $E(\mathcal{G})$ oracle then there should be at least one query for which the result is not repeatable, i.e., the same query gives different outputs on two occasions. We will assume that this result is an independent identically distributed random variable. Now, we forget about everything else, and make the definition of the oracle class $Iid(\delta)$, the Iid referring to *independent identically distributed* random variables. Why forget about everything else? The answer, as we shall see in Theorem 5.3, is

that we do not lose any computational power by doing so.

5.1. The complexity classes BPP and BPP//log⋆

We impose the following conditions on a Turing machine: (a) every step of a computation can be made in exactly two possible ways, which are considered different even if there is no difference in the corresponding actions (this distinction corresponds to two different bit guesses); (b) the machine is clocked by some time constructible function and the number of steps in each computation is exactly the number of steps allowed by the clock – if a final state is reached before this number of steps, then the computation is continued, doing nothing up to this number of steps; (c) every computation ends in a final state, which can be either *accept* or *reject*.

A set A is decided by a *probabilistic Turing machine*, in constructible time t, if, whenever $x \in A$, at least $2^{t(|x|)+1}$ computations (of length $t(|x|)$) end in an accepting state, and, whenever $x \notin A$, strictly less $2^{t(|x|)} + 1$ computations end in an rejecting state.[9]

A set A is decided by a bounded probabilistic Turing machine in constructible time t if there exists a dyadic rational $0 < \varepsilon < \frac{1}{2}$ such that, whenever $x \in A$, the number of computations (of length $t(|x|)$) ending in an accepting state is greater or equal to $(\frac{1}{2} + \varepsilon)2^{t(|x|}$, and, whenever $x \notin A$, the number of computations (of length $t(|x|)$) ending in an rejecting state is greater or equal to $(\frac{1}{2} + \varepsilon)2^{t(|x|}$.[10] BPP is the class of such sets, decided by bounded probabilistic Turing machines in polynomial time.

For the probabilistic complexity classes it is a matter of some controversy whether Definition 4.1 is the appropriate definition of a non-uniform probabilistic class (e.g., of $BPP/\mathcal{F}$). Notice that by demanding that there is a set $B \in BPP$ and a function $f \in \mathcal{F}$ (in this order) such that $w \in A$ if, and only if, $\langle w, f(|w|)\rangle \in B$, we are demanding a fixed probability $\frac{1}{2} + \varepsilon$, $\varepsilon > 0$ (fixed by the Turing machine chosen to witness that $B \in BPP$) *for any possible correct advice*, instead of the more intuitive idea that the error $\gamma = \frac{1}{2} - \varepsilon$ only has to be bounded after choosing the correct advice. This leads to the following definition for the specific complexity class BPP that we will be using in what follows:

Definition 5.1. *$BPP//\log\star$ is the class of sets A for which a probabilistic Turing machine TM, a prefix function $f \in \log\star$, and a constant $\gamma < \frac{1}{2}$ exist such that, for every length n and input w with $|w| \leq n$, TM rejects $\langle w, f(n)\rangle$ with probability at most γ if $w \in A$ and accepts $\langle w, f(n)\rangle$ with probability at most γ if $w \notin A$.*

It is unknown whether $BPP//\log\star \subseteq BPP/\log\star$.

[9] The number of computations is the same, i.e., $2^{t(|x|)}$.
[10] Ibidem.

5.2. Implementing the axioms for Iid(δ).

Here we take the description in Section 2.2, and interpret it in detail for a particular case, called $Iid(\delta)$. Here $\delta \in (0, \frac{1}{2})$ is a given rational number.

Definition 5.2. *The following the axioms define the class of physical oracles* $Iid(\delta)$.

1. Real values. *The experiment is designed to find a physical parameter* $x \in (0, 1)$.
2. Queries. *The query string is a constant.*
3. Finite output. *The result, which we label* X*, is either* 0 *or* 1.
4. Protocol timer. *Each experiment takes place in a fixed time.*
5. Sufficiency of the protocol. *The value of* x *is known to be in* $(\delta, 1 - \delta)$.
6. Repeatability. *The result* X *is an independent identically distributed random variable with probability that* $X = 1$ *being* x.

The first comment is that, to the best of our current physical knowledge, radioactive decay does permit the construction of such an oracle in principle. We could take an atom of a radioactive isotope, for example uranium ^{238}U, place it in a box for a year, and see if it had decayed during that time, the result $X = 1$ being that it had decayed. The physical parameter $x \in (0, 1)$ would be the probability that the atom would decay in one year, which can be calculated from half life of ^{238}U. A parameter $\delta \in (0, \frac{1}{2})$ could be found from a text book value of the half life, about $4{\cdot}46$ billion years, which would come with associated errors, but we could find a value of δ which would be good enough for the experiment. No query is needed, as every experiment would be identical. (It is not difficult to see how to improve on this experiment, but we only want an example!)

The second comment is that we have the following statement of the computational power of this class of oracles, corresponding to Theorem 4.7 for the class $E(\mathrm{ExpLin})$.

Theorem 5.3. *Given a word decision problem in* $BPP//\log\star$*, there is an* $1 > x > 0$ *so that any oracle in class* $Iid(\delta)$ *for real value* $x \in (\delta, 1-\delta)$ *can be used by a Turing machine to solve the problem in polynomial time. Conversely, any decision problem that can be solved in polynomial time by a Turing machine with an oracle in class* $Iid(\delta)$ *is in* $BPP//\log\star$.

We must explain what this class $BPP//\log\star$ is. First, a probabilistic Turing machine is essentially a Turing machine equipped with an oracle which might be called an ideal fair coin toss. Each query leads to the tossing of a fair coin for which

heads and tails are returned with equal probability $\frac{1}{2}$. Each query is probabilistically independent from the other queries. The class $BPP// \log\star$ can be obtained by adding certain advice functions to a probabilistic Turing machine.

Definition 5.4. *$BPP// \log\star$ is the class of sets A for which a probabilistic Turing machine $\mathcal{M}$, a prefix function $f \in \log\star$, and $\gamma < \frac{1}{2}$ exist such that, for every length n and input w with $|w| \leq n$, $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most γ if $w \in A$ and accepts $\langle w, f(n)\rangle$ with probability at most γ if $w \notin A$.*

The next two subsections will be devoted to proving Theorem 5.3, but the reader should note that the proofs are adapted from [5] for the lower bound on computational power, and [7] for the upper bound. In particular, we shall need the following result from [5] to show that $Iid(\delta)$ can simulate the independent coin tosses with probability of heads $\frac{1}{2}$ required in the definition of BPP.

Lemma 5.5. *Take a biased coin (probability of heads $q \in (\delta, 1-\delta)$ for some $\frac{1}{2} > \delta > 0$), and $\gamma \in (0,1)$. Then, up to probability $\geq \gamma$, there is a sequence of independent biased coin tosses of length*

$$\frac{n}{\delta(1-\delta)}\Big(1 + \frac{1}{\sqrt{1-\gamma}}\Big)$$

which gives a sequence of independent fair coin tosses of length n.

After the work of [4], we can safely assume without loss of generality that, for $P/\log\star$ and $BPP// \log\star$, the length of any advice $f(n)$ is exactly $\lfloor a \log_2 n + b \rfloor$, for some $a, b \in \mathbb{N}$ depending on f. The remark above on not losing any computational power by focusing on one experiment with an uncertain result is based on the fact that $P/\log\star \subseteq BPP// \log\star$. This inclusion $P/\log\star \subseteq BPP//\log\star$, is not known to be strict (it is an open question to know whether $P = BPP$). However, it is known that $P \subseteq BPP$ ([2]). It follows, by monotonicity, that $P/\log\star \subseteq BPP//\log\star \subseteq BPP/\log\star$. Notice that, $P/\log \neq P/\log\star$ ([4]).

5.3. Proof of a lower bound for the computational power of Iid(δ).

Take a word decision problem in $BPP// \log\star$. From Definition 5.4, for this particular problem there is an advice function $f(n)$ consisting of taking the first $a\log_2 n + b$ (round down) digits of a particular string y, and a fixed $\gamma < \frac{1}{2}$. Take a rational $\gamma' > 0$ so that $\gamma + 2\gamma' < \frac{1}{2}$. Employ the coding of 4.2 for y, and call the resulting number $x \in (0,1)$ and consider an $Iid(\delta)$ oracle with value $x \in (\delta, 1-\delta)$ for some rational $\delta > 0$. We show that a Turing machine using this oracle of type $Iid(\delta)$ can solve the given word decision problem in polynomial time, with probability of error $\leq \gamma + 2\gamma'$.

There are three processes to consider, firstly determining the required advice, secondly simulating the fair coin toss oracle, and finally running the original program on the Turing machine. Each of these processes has its probability of error, and we shall make these γ', γ' and γ respectively, giving a total probability of error of $\leq \gamma + 2\,\gamma'$.

(1) Determining the required advice: To determine the first k places of x, we need to determine the value of x to within 2^{-k-5}. This is because the distance between x and any dyadic rational with denominator 2^k is at least 2^{-k-5}.

Suppose that the Turing machine queries the $Iid(\delta)$ oracle z times, and keeps count of how many times the result $X = 1$ is obtained (denote this count by L). The value $\tilde{x} = \frac{L}{z}$ will be our estimation of x. As L is a random variable with expected value $\mu = z\,x$ and variance $\nu = z\,x\,(1-x)$, by Chebyshev's inequality we conclude that, for every $\Delta > 0$,

$$\mathbb{P}(|L-\mu| > \Delta) = \mathbb{P}(|z\tilde{x} - z\,x| > \Delta) = \mathbb{P}\left(|\tilde{x} - x| > \frac{\Delta}{z}\right) \leq \frac{\nu}{\Delta^2}.$$

To determine the first k digits of x, we need $|x - \tilde{x}| \leq 2^{-k-5}$, so choosing $\Delta = z\,2^{-k-5}$, we get

$$\mathbb{P}(|\tilde{x} - x| > 2^{-k-5}) \leq \frac{x(1-x)2^{2k+10}}{z} \leq \frac{2^{2k+10}}{z}\ .$$

Thus if we allow a probability $\leq \gamma'$ of making an error in determining the first k digits of x, we need a number z_k of oracle calls given by (up to rounding)

$$z_k \ = \ \frac{2^{2k+10}}{\gamma'}\ .$$

Our problem is now simple – how big does k have to be for an input word of size n? For our problem in $BPP//\log\star$ we require advice of length $a\log_2 n + b$ (round down). Since the coding substitutes three digits for every original digit, we need to read $k = 3\,a\log_2 n + 3\,b$ digits of x. To do this we need z oracle calls, given by (11),

$$z \ = \ \frac{n^{6\,a}\,2^{6b+10}}{\gamma'} \tag{11}$$

(2) Simulating the fair coin toss oracle: For an input of size m, the original program on the Turing machine has polynomial runtime $p(m)$. In particular it can have called the fair coin toss oracle only $p(m)$ times. To simulate this, we construct a

list of $p(m)$ independent fair coin tosses. From Lemma 5.5 we can do this by calling the $Iid(\delta)$ oracle a total of

$$\frac{p(m)}{\delta(1-\delta)}\Big(1+\frac{1}{\sqrt{1-\gamma'}}\Big) \tag{12}$$

times, with probability of error $\leq \gamma'$. One point to remember is that, by Definition 5.4, m is the length of the concatenation encoding $\langle w, f(|w|)\rangle$, where $|w|$ is the length of the input word w, but as m has a linear bound in $|w|$, we still have $p(m)$ having an upper bound that is a polynomial in $|w|$.

(3) Running the original program on the Turing machine: Now we concatenate input word of length n with the advice obtained in stage (1) of length $a\log_2 n + b$, and run the original program. Every time a call of the fair coin toss oracle would be demanded by the original program, we look up the list of pre-prepared simulated fair coin tosses made in (2). This procedure has probability of error $\leq \gamma$, assuming that (1) and (2) were successful.

Summary: An upper bound on the error of the whole procedure can be given by just summing the errors of the constituent parts, i.e. $\gamma + 2\,\gamma'$. Determining the times is a little more complicated. Part (1) requires a number of calls of the $Iid(\delta)$ oracle given by (11), and recording the result. This can be done in time polynomial on n. Part (2) requires a number of calls of the $Iid(\delta)$ oracle given by (12), and running a small program to turn them into fair coin tosses, and recording the result. Part (3) requires a concatenation of the original input word with the advice found in part (1), and then running the original $BPP//\log\star$ program, again in polynomial time given by (12), except for one point. That point is that we have to look up the pre-prepared table of simulated coin tosses instead of just calling a fair coin toss oracle, and this takes a longer time. However consulting this polynomially long list polynomially many times still takes polynomial time. We conclude that the Turing machine with the given $Iid(\delta)$ oracle can solve our $BPP//\log\star$ problem in polynomial time.

5.4. Proof of an upper bound for the computational power of Iid(δ).

We shall follow [7] quite closely. Given a probabilistic oracle which gives two answers (say $X = 1$ and $X = 0$), we can represent the answers to the queries as a binary tree (called a probabilistic tree), where we label each path with its probability. After a certain number of queries, we have accept 'A' or reject 'R' output. An example of this is given in Fig. 2.

To get the probability of acceptance in Fig. 2, we take each descending path from the initial vertex, multiply the probabilities along its edges, and add the results for each path ending in acceptance. Thus the probability of acceptance in Fig. 2 is

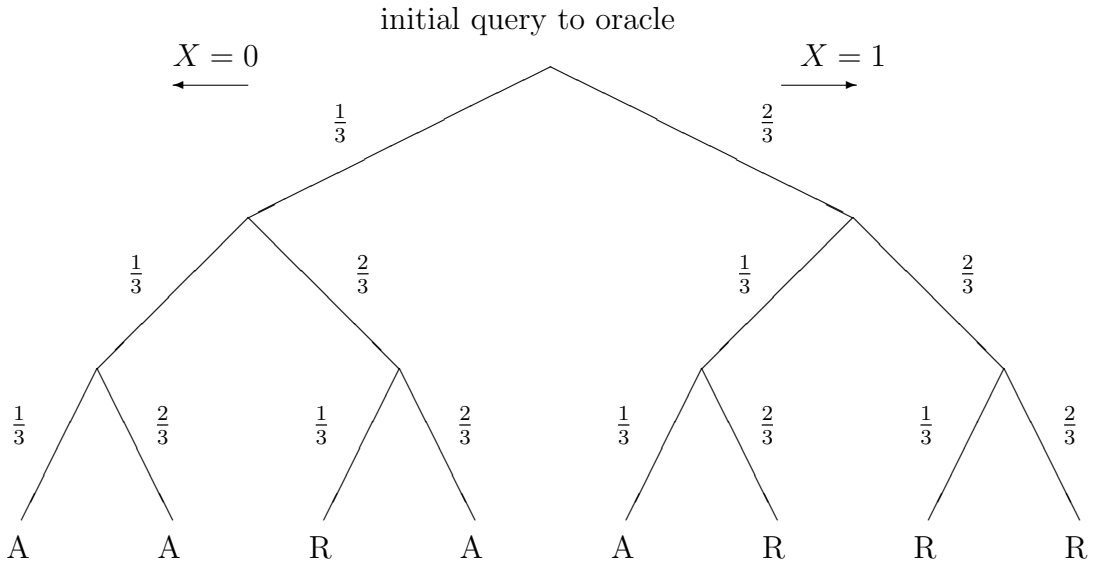

Figure 2. A binary probabilistic tree of height 3, corresponding to 3 queries of a probabilistic oracle

(going through the accepting paths from left to right)

$$\frac{1}{3}\times\frac{1}{3}\times\frac{1}{3} \;+\; \frac{1}{3}\times\frac{1}{3}\times\frac{2}{3} \;+\; \frac{1}{3}\times\frac{2}{3}\times\frac{2}{3} \;+\; \frac{2}{3}\times\frac{1}{3}\times\frac{1}{3}\;.$$

We can ask what would happen if we change the probabilities in the tree, keeping the original Turing machine and tapes. Here it is important to realise that the Turing machine has no knowledge of the probabilities in the tree, it only knows the outcomes. Suppose that we have two probabilistic trees, identical except for the probabilities labelling the edges. The depth of the trees (i.e. the maximum length of a branch descending from the initial vertex) will be m. There is a number k so that for any given edge, the difference in the probabilities for that edge between the two trees is $\leq k$. Then in [7] it is shown that the difference in probability of acceptance for the two trees differs by at most $2mk$. Using this, we can adapt the following Lemma 5.6 from [7]:

Lemma 5.6. *Let $\mathcal{M}$ be a Turing machine with $Iid(\delta)$ oracle for value $x \in (\delta, 1-\delta)$. Suppose that $\mathcal{M}$ decides some set A in time $t(n)$ with error probability bounded by $\gamma < \frac{1}{2}$. Take a rational $\gamma' > 0$ so that $\gamma + \gamma' < \frac{1}{2}$. Let $\mathcal{M}'$ be an identical Turing machine with $Iid(\delta')$ oracle, differing only in its value $x' \in (\delta', 1-\delta')$. If*

$$|x - x'| < \frac{\gamma'}{2\,t(n)}\;,$$

then for any word of size $\leq n$*, the probability of* $\mathcal{M}'$ *making an error when deciding* A *is* $\leq \gamma + \gamma'$*.*

Proof: We set the depth of the tree to $t(n)$, as there may be at most $t(n)$ oracle calls carried out in time $t(n)$. An upper bound to k (the maximum difference in probability labels on the respective edges) is given by the fraction in the statement, so the difference in the probability of acceptance (or similarly rejection) is at most $2\,t(n)\,k$ or γ'. $\square$

Now we can carry out the reverse of Subsection 5.3. There we simulated an advice function by $Iid(\delta)$ oracle calls. Now we simulate $Iid(\delta)$ oracle calls by an advice function and a fair coin toss oracle. This advice will be initial parts of the binary expansion of $x \in (0,1)$. A sufficient length for the advice can be deduced from Lemma 5.6, and we continue with the notation of that Lemma. Suppose that $t(n)$ is a polynomial function of n. Now define $x \restriction m$ to be x rounded down to m binary places x, so for example

$$0{\cdot}101110 \restriction 3 \;=\; 0{\cdot}101 \;.$$

The distance between x and $x \restriction m$ is at most 2^{-m}, so if we set $m_n = \log_2(2\,t(n)/\gamma')$ (round up to integer) and $x' = x \restriction m_n$, then the conditions of Lemma 5.6 are satisfied.

If $t(n)$ is polynomial in n, then m_n is bounded by $a \log_2 n + b$ for some a, b. This means that we can use the first logarithmically long segment of x as advice to construct x', and then, if we have an independent biased coin toss oracle with probability x', we are done. However, we only have a fair (probability $\frac{1}{2}$) coin toss oracle in the definition of $BPP/\!/\log\star$. However we can simulate a biased coin toss oracle with probability x' using a fair coin toss oracle. As x' is a rational with denominator 2^m, we can generate a random number using m fair coin tosses. Each toss gives a zero or one in an integer. Then compare this to the fraction x' to determine whether the biased coin toss should be a head. For example, suppose that $x' = {\cdot}11$ (the rational $\frac{3}{4}$), so $m = 2$. The integers of length 2 from the fair coin toss oracle are (with equal probability) 00, 01, 10 and 11. Taking the expansion of x', we have a head if the integer is < 11, so we have a $\frac{3}{4}$ chance of a head. This completes the proof.

6. Conclusion

Our axioms for interfacing physical systems and algorithms are quite abstract. There are many experiments that satisfy the axioms and, as our earlier detailed analysis of particular physical experiments have established [5, 7, 9, 10, 12, 8], these

axioms are close to our theoretical understanding of reality. To reflect on their physical interpretation, consider the question:

What does the computational power of algorithms boosted by oracles that are physical systems tell us about the physical systems?

The computational power of a Turing machine augmented by an oracle benchmarks how useful the oracle's information can be in boosting a computation; hopefully, it provides some precise method of quantifying the information provided by the physical experiment. Given our axiomatic characterisations, and the fact that their computational power transcends the Turing barrier, we consider the questions:

How much information can be extracted from a physical process? How quickly can it be obtained?

6.1. Physical interpretations of observing experiments

Consider the time bounds we discovered in particular experiments.

Deterministic systems. A time bound, such as the following inequality,

$$\frac{A}{|x-y|^q} \;\leq\; T_{\text{experiment}}(x,y) \;\leq\; \frac{B}{|x-y|^n}\;, \tag{13}$$

where $n, q \geq 1$ and $A, B > 0$ are constants determined by the experiment, has an implication for the time taken to extract information from the 'universe'. But the principle is simple – if we consider the binary expansion of the real world value $x = 0{\cdot}x_1x_2x_3\ldots$ (for example), determining the binary digit x_n takes an amount of time at least exponential in n. In a certain sense, *the more information we have, the more difficult it is to extract new information.*

How relevant are exponential runtimes (13) to the work of an observer? The first point is that measuring finitely many quantities does not really alter (13). If the experimenter measures 15 quantities, and each obeys a version of (13), then we can plot the results on $\mathbb{R}^{15}$ by treating each point as a coordinate. Then we can take x and y as points of $\mathbb{R}^{15}$, and use the standard distance on $\mathbb{R}^{15}$, and get the same form for the time taken.

Where we might see a difference is where the observer can measure a potentially infinite number of quantities. Consider, for example, an astronomer who measures the brightness (absolute luminosity) of one star very accurately, it might be reasonable to expect (13) to apply. By the previous paragraph, if an astronomer measures the brightness of 24,789 stars very accurately, it might be reasonable to expect (13) to apply. However the astronomer might measure the brightness of a different star every day, but to a certain fixed accuracy. In principle this seems to allow for extraction of information from the universe at a *constant* rate. However, there are at

least two problems with this view.

(i) As more stars are observed, the observer is reduced to measuring more distant stars, and it is more difficult to measure the brightness of these fainter stars to the degree necessary to establish their absolute luminosity (indeed, it becomes more difficult to observe them at all). It is likely that similar problems could be raised with any observation of 'arbitrarily many' quantities.

(ii) The other problem is that as measurements are made, it may become more obvious that the brightness of stars is falling into a predictable pattern, in other words a physical law may be postulated, and the variation allowed for within this law may be essentially random. This means that the information may be less useful than it seems at first, as it is a mixture of random variation and predictable information.

This brings us to two cases where observing a localised system might be considered to cause problems with our ideas of information.

Random systems. The first is the case of an inherently random system (see [14]). An independent sequence of coin tosses (or similar statistical information) really can assist a computation, in terms of complexity theory. Further, according to our current interpretation of quantum mechanics, such sources of information really do occur in nature, for example radioactive decay. In (13) we considered a deterministic experiment – there is no doubt about the outcome, just the matter of how long it takes to obtain it. However in several papers (see [5] or [7]) we considered probabilistic outcomes. Instead of continually improving the accuracy of an experiment, we ran an identical experiment many times, and used randomness and averaging to do the work of finding the desired information about the real world quantity. As long as we work in the established framework of complexity theory allowing for a certain probability of error, we found the same complexity theory results from such a probabilistic machine as from the non-probabilistic experimental setup giving (13). (To be precise, we should also add an independent coin toss machine to the non-probabilistic machine.) So far from being a problem, the probabilistic approach adds weight to the non-probabilistic oracle results, as these seemingly totally different approaches to experimentation produce the same results in complexity theory.

Chaotic systems. The most problematic classical observations, from the point of causing trouble with our conjecture, seem to be observations of chaotic systems. In principle, a real world initial condition $x = 0 \cdot x_1 x_2 x_3 \ldots$ might be multiplied by an exponential factor in time, and we might read the binary digits x_n in a time linear in n. As an example, we could take the double and then fractional part map from

$[0, 1)$ to itself:

$$f(x) \;=\; \begin{cases} 2\,x & x < 1/2\ , \\ 2\,x - 1 & x > 1/2\ . \end{cases}$$

However this map is not time reversible, which is at variance with at least some of the philosophy of modern physics. If a chaotic system is reversible, and has a bounded phase space (a reasonably realistic condition) then such exponential growth could not be maintained forever. In this case it is not so obvious what measurements could be made to determine the digits of x in constant time. In any case, a real world chaotic system suffers from another problem, which would make make determination of x rather difficult. Random fluctuations, from whatever cause, are also magnified exponentially, and the net result is that, no matter what the initial conditions, the system is likely to become completely random for large enough time.

6.2. Scope and limits to measurement

There is a role for the axioms to explore the scope and limits to measurement.

Invariance and Portability. In general, the purpose of a set of axioms is to isolate those essential properties that are both fundamental and common to a perceived or imagined class of systems. Later, the axioms come to define the class. The axioms bring with them the notions of an interface and models of its many possible implementations. Our axioms can be applied to a rather extraordinary range of experiments. However, the theorems show that:

As far as the algorithm is concerned, the experiments satisfying the axioms have the same computational power, whatever their underlying science or technology.

This is a physical idea similar to the notion of *portability* in software: programs need to run on different machines with different architectures and technologies.

Non-determinism. The experiments specified in [5, 7, 9, 10, 12, 8] have a simple form yet they are found throughout science and engineering and provide a canonical type for the logical theory of measurement (see [8]). The sets of mathematical axioms are very similar but it is important to notice the fundamental role of the last property, *repeatability*. First, the axiom used to prove upper bounds needs to be stronger than that for lower bounds in the case of exact and arbitrary precision.

Second, in our treatment of fixed precision, the repeatability axiom changes the theorems considerably: the use of a probability distribution introduces non-deterministic behaviour. The axioms lead us this general observation:

Imprecision and, therefore, uncertainty and non-determinism are ubiquitous features of classical physics, once assumptions about upper bounds on accuracy or fixed

error margins are employed in their description.
Thus, at first sight, even in the simplest of physical situations, it may not be obvious whether apparent non-determinism in the behaviour of a system is intrinsic to the system, or caused by our measurement process.

Turing Barrier. The axioms apply to many physical quantities, belonging to many physical theories, measured by means of many forms of equipment that are constructed from many technologies. The termination of algorithms is a fundamental problem that has been analysed in many ways (using hierarchies, degrees, proof systems and their ordinals). Consider two versions of the halting problem for Turing machines:

$Halt = \{(T, x)|$ Turing machine T halts on input $x\}$

and

$Total = \{T|$ Turing machine T halts on all inputs $x\}$.

It is easy to reduce algorithmically $Halt$ to $Total$ but not *vice versa*.[11] Formulations of the set $Total$ exist in $P/\log\star$ (e.g., one based upon the tally set of unary encodings of Turing machines). Since the total halting problem for Turing machines lies in $P/\log\star$, the axioms demonstrate that it is common for physical measurements to transcend the algorithmic limits proposed by the Church-Turing Thesis of 1936. The computational classification of the class $P/\log\star$ may lead to a taxonomy of experimental methods.

References

[1] T. P. Baker, J. Gill, and R. Solovay. Relativizations of the $P =?NP$ question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[2] José Luis Balcázar, Josep Días, and Joaquim Gabarró. *Structural Complexity I.* Springer-Verlag, 2nd edition, 1988, 1995.

[3] José Luis Balcázar, Josep Días, and Joaquim Gabarró. *Structural Complexity II.* Springer-Verlag, 1990.

[4] José Luis Balcázar and Montserrat Hermo. The structure of logarithmic advice complexity classes. *Theoretical Computer Science*, 207(1):217–244, 1998.

[5] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Computational complexity with experiments as oracles. *Proceedings of the Royal Society,*

[11] The set $Total$ is complete Π_2 and the set $Halt$ is complete Σ_1.

Series A (Mathematical, Physical and Engineering Sciences), 464(2098):2777–2801, 2008.

[6] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Oracles and advice as measurements. In Cristian S. Calude, José Félix Costa, Rudolf Freund, Marion Oswald, and Grzegorz Rozenberg, editors, *Unconventional Computation (UC 2008)*, volume 5204 of *Lecture Notes in Computer Science*, pages 33–50. Springer-Verlag, 2008.

[7] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Computational complexity with experiments as oracles II. Upper bounds. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 465(2105):1453–1465, 2009.

[8] Edwin Beggs, José Félix Costa, and John V. Tucker. Computational Models of Measurement and Hempel's Axiomatization. In Arturo Carsetti, editor, *Causality, Meaningful Complexity and Knowledge Construction*, volume 46 of *Theory and Decision Library A*, pages 155–184. Springer, 2010.

[9] Edwin Beggs, José Félix Costa, and John V. Tucker. Limits to measurement in experiments governed by algorithms. *Mathematical Structures in Computer Science*, 20(06):1019–1050, 2010. Special issue on Quantum Algorithms, Editor Salvador Elías Venegas-Andraca.

[10] Edwin Beggs, José Félix Costa, and John V. Tucker. Physical oracles: The Turing machine and the Wheatstone bridge. *Studia Logica*, 95:271–292, 2010. Special issue: The contributions of Logic to the Foundations of Physics, Editors D. Aerts, S. Smets & J. P. Van Bendegem.

[11] Edwin Beggs, José Félix Costa, and John V. Tucker. Unifying science through computation: Reflections on computability and physics. In Olga Pombo, Juan Manuel Torres, John Symons, and S. Rahman, editors, *New Approaches to the Unity of Science, Vol. II: Special Sciences and the Unity of Science*, volume 24 of *Logic, Epistemology, and the Unity of Science*, pages 53–80. Springer, 2012.

[12] Edwin Beggs, José Félix Costa, and John V. Tucker. The impact of limits of computation on a physical experiment. *Mathematical Structures in Computer Science*, in press. Special issue on Computability of the Physical, Editors Cristian S. Calude and S. Barry Cooper.

[13] Edwin Beggs and John V. Tucker. Experimental computation of real numbers by Newtonian machines. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 463(2082):1541–1561, 2007.

[14] Cristian Calude. Algorithmic randomness, quantum physics, and incompleteness. In Maurice Margenstern, editor, *Machines, Computations and Universality (MCU 2004)*, volume 3354 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2004.

[15] Richard M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[16] Albert Meyer and Larry Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proceedings of the 13th. IEEE Symp. on Switching and Automata Theory*, pages 125–129. IEEE, 1973.

[17] Larry Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1977.

[18] John V. Tucker and Jeffery I. Zucker. Computable functions and semicomputable sets on many sorted algebras. In Samson Abramsky, Dov Gabbay, and Tom Maibaum, editors, *Handbook of Logic for Computer Science*, volume V, pages 317–523. Oxford University Press, 2000.

[19] John V. Tucker and Jeffery I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational Logic*, 5:611–668, 2004.

[20] John V. Tucker and Jeffery I. Zucker. Computable total functions, algebraic specifications and dynamical systems. *Journal of Algebraic and Logic Programming*, 62:71–108, 2005.

[21] Alan Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45:161–228, 1939.

[22] Celia Wrathall. Complete sets and the polynomial time hierarchy. *Theoretical Computer Science*, 3(1):23–33, 1977.