

The Third Kind of Step

What Every Proof of P ≠ NP and BQP ≠ BPP Must Contain

Forrest Landry

UVSM Publications
Wytheville, Virginia

In three parts.
Part I: The Two Separations.
Part II: The Template.
Part III: The Two Classifications.

The Third Kind of Step; What Every Proof of P ≠ NP and BQP ≠ BPP Must Contain

© 2026 Forrest Landry.
Some rights reserved.

Published by UVSM Publications LLC
965 Reed Creek Mill Rd
Wytheville, VA 24382-8287
USA

uvsmpublications.com
contact@uvsmpublications.com

ISBN 979-8-90710-083-1
DOI: https://doi.org/10.67291/UKTZ6086

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

You may share this work for non-commercial purposes with proper attribution, but you may not adapt, transform, or build upon it.

Full license text: https://creativecommons.org/licenses/by-nc-nd/4.0/

Author ORCID: 0009-0005-6275-0362

The license above permits sharing this work for non-commercial purposes with attribution. Beyond what it permits, no part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without prior written permission of the publisher.

Contents

Introduction

Part I. The Two Separations

1. The Question
2. The Classes
3. The Evidence
4. The Three Barriers
5. What the Barriers Share

Part II. The Template

6. The Kinds of Step
7. What a Proof Must Do and What It Must Assume
8. The Template for a Separation

Part III. The Two Classifications

9. BQP ≠ BPP: What It Asserts and What Every Proof of It Must Show
10. BQP ≠ BPP: The Third Kind
11. P ≠ NP: What It Asserts and What Every Proof of It Must Show
12. P ≠ NP: The Third Kind
13. The Record
14. Questions and Answers
15. What the Book Establishes

Appendices

Appendix A. The Barrier Theorems: Proof Sketches

Appendix B. The Three Kinds as the Sectors of the Comparison Algebra

Glossary

Bibliography

Index

Introduction

The Question

Two of the central separation claims of the theory of computation have the same shape. The first, P $\neq$ NP, says that some problem whose answers can be checked quickly by a deterministic machine cannot be solved quickly by one. The second, BQP $\neq$ BPP, says that some problem that a quantum computer solves quickly, with bounded error, cannot be solved quickly by a classical machine that flips coins. Each compares two models of computation that differ in one respect only, the way the machine branches, and each says that the difference changes what the machine can do in polynomial time. For both, three barrier theorems, of the same form and each proved for its own pair of classes, show that the familiar methods of proof cannot settle them.

This book asks what every proof of such a separation must contain. The question is about the proofs, not about the answer. The book does not prove either separation, and it does not show that either has no proof. It shows, under a premise it names, that every proof of either separation must contain a step of a definite kind: a step that derives what a model of computation can do from what the model is, or a limit on what every machine of a class can do from what those machines are. Every technique that takes no such step relativizes, and so, by the oldest of the three barrier theorems, cannot settle either separation; the natural and algebrizing techniques take such a step, in forms that their own barrier theorems exclude.

The Answer

The answer uses a framework developed in the companion book *Formalizability, Independence, Soundness*, here called the FIS book. Every step of a formal proof is of one or more of three kinds. A step of the first kind gives a verdict that does not depend on the order of its terms: checking identity, substituting equals for equals, applying a definition, counting. A step of the second kind relates two terms in roles that cannot be exchanged: case analysis whose

cases are handled by different arguments, an oracle query and its answer, a challenge and its response. A step of the third kind derives what a process does from what the process is, or a limit on what the members of a class of processes can do from what those members are. A proposition is a third-kind proposition when every formal, sound proof of it that does not assume it openly uses a step of the third kind about the processes the proposition is about: a step that derives, from what those processes are, what they do or what they cannot do, where that doing is part of what the proposition states.

The book gives one template for showing that a separation of two complexity classes is a third-kind proposition. The template has three inputs: the relativization barrier for the pair of classes, which shows that no relativizing technique settles the separation; the covering lemma, which the book proves, and which shows that every argument confined to the first two kinds of step relativizes; and a step premise, which states that the step of the third kind each proof then contains derives, from what they are, what the machines of the stronger model can do or what the machines of the weaker model cannot do. From these, a short lemma gives the classification. The other two barrier theorems, those of natural proofs and of algebrization, show which steps of the third kind are not enough.

The book applies the template to both separations. For BQP $\neq$ BPP the barrier theorems are those of Bernstein and Vazirani, Simon, Razborov and Rudich, and Aaronson and Wigderson, stated for the quantum and classical classes. For P $\neq$ NP they are those of Baker, Gill and Solovay, Razborov and Rudich, and Aaronson and Wigderson. Under the step premise, the template gives the same result for both: each separation is a third-kind proposition.

What the Result Is About

The result says what every proof of a separation must do. It is a different thing from what every proof must assume, and the book keeps the two apart. The FIS book proves that a proposition has no proof that is at once formal, independent and sound when every formal derivation of it must assume part of what it claims. A proof that derives what a machine can do from what the machine is assumes nothing about its conclusion: its premises can be the definitions of the machines, and nothing more. So the third-kind classification

does not apply the FIS Theorem to either separation, and the book does not claim that either separation lacks a proof with all three properties. It states what such a proof would contain.

Each separation has a proven counterpart of smaller scale that shows this. Bravyi, Gosset and König proved, without any assumption, that constant-depth quantum circuits solve a problem that constant-depth classical circuits with gates of bounded fan-in cannot. Paul, Pippenger, Szemerédi and Trotter proved that nondeterministic linear time is strictly more powerful than deterministic linear time. Each proof derives a capacity of one model and a limit of the other from what the two models are, and each is formal, independent and sound.

The Structure of the Book

The book has three parts.

Part I, The Two Separations (Chapters 1 to 5), states the question and its ground. Chapter 1 states the question. Chapter 2 defines the classes P, NP, BPP and BQP, the quantum model of computation, and Boolean circuits. Chapter 3 sets out the evidence for each separation. Chapter 4 states the three barrier theorems for both separations. Chapter 5 states what the three barriers have in common.

Part II, The Template (Chapters 6 to 8), states the framework. Chapter 6 imports the three kinds of step, the required engagement of a proposition and the sector-elimination lemma from the FIS book, and fixes the test that separates the first kind of step from the third. Chapter 7 imports the three properties of proof and the FIS Theorem, and states why the third-kind question and the question of circularity are different questions. Chapter 8 proves the covering lemma, and gives and proves the template for a separation of two complexity classes.

Part III, The Two Classifications (Chapters 9 to 15), applies the template. Chapters 9 and 10 apply it to BQP $\neq$ BPP, and Chapters 11 and 12 to P $\neq$ NP. Chapter 13 records the results that bear on each separation. Chapter 14 answers the questions readers ask, and Chapter 15 states what the book establishes.

Appendix A gives proof sketches of the barrier theorems. Appendix B sets out, from the FIS book, that the three kinds of step are the three sectors of the comparison algebra. That result shows where the three kinds come from; the results of this book are proved without it.

Self-Containment

The book imports what it needs from the FIS book and states each imported definition and result in full where it is used, with its number in that book. The barrier theorems are results of complexity theory, cited from their authors and sketched in Appendix A. The premise of the two classifications is named where it is used, and every ground they rest on is collected in Chapter 15.

Intended Audience

The book is written for readers who know the elements of computational complexity: Turing machines, polynomial time, the classes P and NP, and probabilistic computation. Chapter 2 states every definition the later chapters use, including those of quantum computation, so a reader who knows the classical classes can follow the quantum material from that chapter.

PART I

The Two Separations

Chapter 1. The Question

ABSTRACT

P ≠ NP and BQP ≠ BPP each say that two models of computation, alike in every resource and different in how they branch, can do different things in polynomial time. Three theorems show that the techniques of relativization, natural proofs and algebrization cannot give a proof of either. This chapter separates three questions that can be asked about such a separation: whether it is true, what every proof of it must contain, and what every proof of it must assume. I answer the second question for both separations, under a premise I name, and I state how that answer differs from an answer to the third question.

1.1. Two Separations

A deterministic machine follows one path of computation. A nondeterministic machine accepts an input when some path accepts it, and so it can be described as a deterministic machine that checks a proposed witness. The class P contains the decision problems that a deterministic machine solves in polynomial time; the class NP contains those whose yes answers have witnesses that a deterministic machine checks in polynomial time. The separation P ≠ NP says that some problem in NP is not in P: that checking a witness is, for some problems, easier than finding one.

A classical probabilistic machine flips fair coins and answers correctly with probability at least two thirds. A quantum machine applies unitary transformations to a register of quantum bits and then measures the register, and the quantum machine too answers correctly with probability at least two thirds. The class BPP contains the problems a probabilistic machine solves in polynomial time, and BQP those a quantum machine solves in polynomial time. The separation BQP ≠ BPP says that some problem in BQP is not in BPP: that for some problems, a quantum computer is faster than any classical randomized one by more than a polynomial factor.

The two separations share a shape. In each, the two classes have the same bound on time, polynomial in the length of the input, and the same notion of a correct answer. They differ in the way the machine branches: deterministically or existentially for P and NP, by independent coin flips or by the interference of complex amplitudes for BPP and BQP. Each separation says that the difference in the way the machine branches changes the class of problems that the machine can solve in polynomial time.

Neither separation is known to imply the other. P $\neq$ NP was posed by Cook in 1971 and by Levin in 1973; BQP $\neq$ BPP became a precise question when Bernstein and Vazirani defined BQP in 1993. Each is supported by a large body of evidence, set out in Chapter 3.

1.2. Three Theorems About the Proofs

For both separations, three theorems show that whole families of proof techniques cannot settle them.

The first of the three barrier theorems is the relativization barrier. A technique relativizes when its argument goes through unchanged if every machine in it is given access to the same arbitrary oracle. Baker, Gill and Solovay (1975) gave oracles relative to which P equals NP and oracles relative to which they differ. Bernstein and Vazirani (1997) and Simon (1997) gave oracles relative to which BQP and BPP differ, and an oracle for a problem complete for polynomial space makes them equal. So no relativizing technique settles either question.

The second of the three barrier theorems is the natural proofs barrier of Razborov and Rudich (1997). A natural proof shows that a function needs large circuits by exhibiting a property of truth tables that is cheap to test, held by many functions, and held by no function with small circuits. If strong pseudorandom function generators exist, no natural proof shows that any function needs superpolynomial circuits, and that blocks the route to either separation through a circuit lower bound.

The third of the three barrier theorems is the algebrization barrier of Aaronson and Wigderson (2009). A technique algebrizes when its argument still holds after an oracle is paired with a low-degree polynomial extension of

it, the extension given to the class that must be shown the stronger. Aaronson and Wigderson gave pairs of oracles and extensions on both sides of both questions, so no algebrizing technique settles either one.

Chapter 4 states the three barriers exactly for both separations. Together the three barriers cover diagonalization and simulation, the combinatorial lower bounds against restricted circuits, and arithmetization, the method behind the theorem that interactive proofs decide exactly the problems solvable in polynomial memory.

1.3. Three Questions

Three different questions can be asked about such a separation.

The first question is whether the separation is true. I do not answer the first question for either separation.

The second question is what every proof of the separation must contain. A barrier theorem answers part of this question: it says that a proof of the separation cannot be a member of a certain family. The question can be asked more sharply. Is there a kind of reasoning that every proof of the separation must use, and that the techniques the barriers cover do not use? I answer that question for both separations, under a premise I name.

The third question is what every proof of the separation must assume. A proof can establish a true conclusion and still take part of that conclusion as given, openly as a premise, or silently in the conditions it must meet to count as a proof. The FIS book shows that for some propositions every formal derivation must do this, and that such a proposition has no proof that is at once formal, independent and sound. I do not show in this book that either separation is a proposition of that kind, for which every formal derivation must take part of the proposition as given.

The second and third questions are easy to run together. A proof that must derive what a process does from what the process is, when its conclusion is about exactly that, may look as if it assumes its conclusion. It does not. A proof that a given program sorts every list reasons from the program's text to its behavior, which is the relation its conclusion states, and its premises are

the text and the rules of the language, nothing more. Reasoning about a relation is not assuming it. I answer the second question, and I state at each point what the answer does not say about the third question.

1.4. The Answer in Brief

The answer has three parts.

First, every step of a formal proof is of one or more of three kinds, defined in the FIS book. The first kind compares values without regard to order. The second relates two terms in roles that cannot be exchanged. The third derives what a process does from what it is, or a limit on what a class of processes can do from what its members are. Chapter 6 states the definitions.

Second, every argument confined to the first two kinds of step relativizes. An argument of the first two kinds uses a machine only as a computation: it runs the machine, counts its steps, builds objects from its description and branches on its outputs, and every one of these goes through unchanged when every machine is given the same oracle. Chapter 8 proves this as the covering lemma. A relativizing argument uses each oracle answer as a value fed to one argument, which is the first kind. A natural proof takes a step of the third kind in showing that small circuits lack its property, and then uses that result only through a cheap test on truth tables. An algebrizing argument takes a step of the third kind through the completeness results it cites, which rest on the fact that each step of a machine changes only a small window of its tape, and then uses that step's result only in the arithmetized form that a low-degree extension preserves.

Third, a short lemma turns the failure of relativizing techniques into a statement about every proof. Since every technique confined to the first two kinds relativizes, and no relativizing technique settles either separation, every proof of either separation has a step of the third kind. If that step derives what the machines the separation compares can or cannot do, the separation is a third-kind proposition. The condition that the step derives what the machines compared can or cannot do is stated as a premise, the step premise, Premise L. Chapter 8 proves the covering lemma and the template, and Chapters 10 and 12 apply the template to the two separations.

The result says more than that proofs of separations are about machines. A proof can derive what machines can or cannot do by steps of the first kind alone, as the proof of the time hierarchy theorem does (Section 8.2); the result is that no proof of either separation can.

1.5. What the Answer Explains

The result explains why the barriers have the shape they have. They are not three unrelated obstacles. Relativization blocks every technique that takes only steps of the first two kinds; the natural proofs and algebrization barriers block two forms of the third kind; and each separation needs a step of the third kind of a form that none of the three blocks. A proof of either separation must, somewhere, derive what the stronger model can do or what the weaker model cannot do from what the models are, in a step of a form the eliminated techniques do not take.

The result also says where a proof can be found. Williams's algorithmic method, which derives a circuit lower bound from a fast algorithm for the satisfiability of circuits of a class, takes a step of the third kind: it reasons from what the circuits of a class are to what can be decided about them. Williams's method lies outside the three families, and the method has proved lower bounds that no technique in the three families had reached.

1.6. Summary

P $\neq$ NP and BQP $\neq$ BPP each say that the way a machine branches changes what it can do in polynomial time. Three theorems show that relativizing, natural and algebrizing techniques settle neither. I ask what every proof of such a separation must contain, which is a different question from whether the separation is true and from what every proof of it must assume. My answer, under one named premise, is that every proof of either separation contains a step of the third kind that derives, from what the machines are, what the machines of one model can do or what those of the other cannot do. Chapter 2 defines the classes the two separations compare.

Chapter 2. The Classes

ABSTRACT

This chapter defines the objects the two separations are about. It defines the Turing machine and its deterministic, nondeterministic and probabilistic variants; the classes P, NP and BPP; the quantum model of computation and the class BQP; Boolean circuits and the class P/poly; and the polynomial hierarchy. In this chapter I also state the known containments among these classes, and what is known about how the two separations relate. I develop only what the later chapters use.

2.1. The Turing Machine

Definition 2.1 (Turing machine). A Turing machine is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, where Q is a finite set of states; Σ is a finite input alphabet, not containing the blank symbol; Γ is a finite tape alphabet with $\Sigma \subset \Gamma$ and the blank symbol in Γ; $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function; q_0 is the start state; and q_{acc} and q_{rej} are the accept and reject states, which are different.

At each step the machine reads the symbol under its head, writes a symbol, moves one cell left or right, and enters a new state, as the transition function directs. The machine halts when it enters the accept or the reject state. The transition function is the whole specification of the machine, and it is a finite table. A machine runs in time $T(n)$ when it halts within $T(n)$ steps on every input of length n, and in polynomial time when $T(n)$ is at most n^k for some constant k. Every reasonable deterministic model of computation simulates every other with at most polynomial overhead, so the class of problems solvable in polynomial time does not depend on the model.

2.2. Three Ways to Branch

Definition 2.2 (Nondeterministic Turing machine). A nondeterministic Turing machine is a Turing machine whose transition function is replaced by a transition relation, so that a state and a symbol can have several possible transitions. It accepts an input when some sequence of transitions from the starting configuration reaches the accept state.

Definition 2.3 (Probabilistic Turing machine). A probabilistic Turing machine is a Turing machine whose transition function gives, for each state and symbol, two transitions, each taken with probability one half. Its probability of accepting an input is the fraction of its runs, over all sequences of choices, that reach the accept state.

The three machines branch in three ways. A deterministic machine follows one path. A nondeterministic machine accepts when some path accepts: its branching is resolved by existence. A probabilistic machine follows one path chosen by its coin flips, and its answer is the answer of most paths, weighted equally. Its branches are independent, and they combine only by counting.

2.3. The Classical Classes

Definition 2.4 (P). A language L is in P when some deterministic Turing machine decides L in polynomial time.

Definition 2.5 (NP). A language L is in NP when there are a deterministic polynomial-time machine V, the verifier, and a polynomial q such that for every input x: x is in L exactly when some string w of length at most $q(|x|)$ makes V accept (x, w). The string w is a witness.

The two definitions of NP agree: a nondeterministic machine running in polynomial time accepts exactly the inputs for which some accepting path exists, and a description of that path is a witness that a deterministic machine

checks in polynomial time. A language is NP-complete when it is in NP and every language in NP reduces to it in polynomial time. Cook (1971) and Levin (1973) showed that satisfiability of Boolean formulas, SAT, is NP-complete, so P $\neq$ NP holds exactly when SAT is not in P.

Definition 2.6 (BPP). A language L is in BPP when some probabilistic Turing machine M runs in polynomial time and, for every input x, accepts with probability at least $2/3$ when x is in L and rejects with probability at least $2/3$ when x is not in L.

The bound of $2/3$ can be replaced by any constant strictly between $1/2$ and 1. Running the machine k times and taking the majority answer brings the probability of error below $2^{-\Omega(k)}$, by the Chernoff bound, so the class does not change. It is widely believed that BPP = P; Impagliazzo and Wigderson (1997) showed that this follows if some problem solvable in time $2^{O(n)}$ needs circuits of size $2^{\Omega(n)}$.

2.4. Quantum Computation

A quantum computer operates on quantum bits. The state of n quantum bits is a unit vector in a complex vector space of dimension 2^n, with one coordinate, an amplitude, for each of the 2^n strings of n bits. A computation applies unitary transformations, called gates, each acting on one or two quantum bits, and ends with a measurement, which returns the string s with probability equal to the squared magnitude of the amplitude of s.

The difference from probabilistic computation can be stated exactly. A probabilistic computation on n bits has a state that is a vector of 2^n probabilities, nonnegative real numbers that sum to one, and each step multiplies it by a stochastic matrix, whose entries are nonnegative and whose columns sum to one. A quantum computation has a state that is a vector of 2^n amplitudes, complex numbers whose squared magnitudes sum to one, and each step multiplies it by a unitary matrix, whose entries may be negative or complex. When two paths of a probabilistic computation reach the same outcome, their probabilities add, and the sum is at least as large as each. When two paths of

a quantum computation reach the same outcome, their amplitudes add, and they can cancel: paths with amplitudes $1/\sqrt{2}$ and $-1/\sqrt{2}$ give that outcome probability zero. This cancellation is called destructive interference, and its counterpart, constructive interference, is the reinforcement of amplitudes with the same phase. Quantum algorithms are designed so that the paths to wrong answers cancel and the paths to the right answer reinforce.

Definition 2.7 (Quantum circuit). A quantum circuit on n quantum bits is a sequence of unitary gates $U_1, \ldots, U_m$, each acting on at most two quantum bits, followed by a measurement of one or more quantum bits. Its size is m, and its depth is the number of layers of gates, where gates on disjoint quantum bits share a layer.

The gates Hadamard, phase, controlled-NOT and the $\pi/8$ gate form a universal set: every unitary on a fixed number of quantum bits is approximated to any precision ε by a circuit of these gates, and by the Solovay–Kitaev theorem (Kitaev, 1997) the circuit needs only a number of gates polylogarithmic in $1/\varepsilon$. So the class defined next does not depend on the universal set chosen.

Definition 2.8 (BQP). A language L is in BQP when there is a uniform family of polynomial-size quantum circuits $\{C_n\}$, uniform meaning that a deterministic machine outputs C_n in time polynomial in n, such that for every input x of length n: measuring the output quantum bit of C_n on input x gives 1 with probability at least $2/3$ when x is in L, and 0 with probability at least $2/3$ when x is not in L (Bernstein and Vazirani, 1997).

Uniformity keeps the circuit family from carrying information that no algorithm could produce. As with BPP, the bound $2/3$ can be raised as close to 1 as desired by repetition.

2.5. Boolean Circuits and P/poly

Definition 2.9 (Boolean circuit; P/poly). A Boolean circuit on n inputs is a directed acyclic graph whose sources are the inputs and whose other nodes are AND, OR and NOT gates, with one or more designated outputs. Its size is the number of gates. A language is in P/poly when there is a polynomial p and, for each n, a circuit of size at most $p(n)$ that decides the language on inputs of length n.

Circuits are non-uniform: a different circuit may be used for each length of input, and nothing requires that an algorithm produce them. So P/poly contains P, and it also contains undecidable languages. Adleman (1978) showed, for machines with one-sided error, and Bennett and Gill (1981) for BPP by the same argument, that BPP is contained in P/poly: for each input length some fixed sequence of coin flips gives the right answer on every input of that length, and it can be built into the circuit.

Circuit lower bounds give routes to both separations. A language in NP that is not in P/poly is not in P, so a superpolynomial circuit lower bound for a problem in NP gives P $\neq$ NP. A language in BQP that is not in P/poly is not in BPP, by the theorem of Adleman, Bennett and Gill, so a superpolynomial circuit lower bound for a problem in BQP gives BQP $\neq$ BPP. Neither separation is known to require such a lower bound.

2.6. The Polynomial Hierarchy

Definition 2.10 (The polynomial hierarchy). Let ${\Sigma_0}^{\mathrm{P}} = {\Pi_0}^{\mathrm{P}} = \mathrm{P}$. For $k \geq 0$, ${\Sigma_{k+1}}^{\mathrm{P}}$ is the class of languages of the form $\{x : \text{some } y \text{ of polynomial length has } (x, y) \text{ in } L\}$ with L in ${\Pi_k}^{\mathrm{P}}$, and ${\Pi_{k+1}}^{\mathrm{P}}$ the class of languages of the form $\{x : \text{every } y \text{ of polynomial length has } (x, y) \text{ in } L\}$ with L in ${\Sigma_k}^{\mathrm{P}}$. PH is the union of all the levels.

NP is Σ_1^{P}. If two adjacent levels coincide, the whole hierarchy collapses to that level; a collapse is widely regarded as implausible, and several results below are stated as consequences of the hierarchy not collapsing.

2.7. The Landscape

Two larger classes bound from above the classes defined so far. PSPACE is the class of problems a deterministic machine decides using an amount of memory polynomial in the length of the input, with no limit on its time. NEXP is the class of problems a nondeterministic machine decides in time 2 raised to a polynomial in the length of the input. The known containments are these:

$$\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PSPACE}, \qquad \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE}.$$

Each containment has a short reason. A deterministic machine is a probabilistic machine that ignores its coins, so $\mathrm{P} \subseteq \mathrm{BPP}$. A quantum circuit can prepare a quantum bit in an equal superposition and measure it, which is a fair coin, so $\mathrm{BPP} \subseteq \mathrm{BQP}$. A quantum computation can be simulated in polynomial space by computing each amplitude as a sum over the paths of the computation, one path at a time (Bernstein and Vazirani, 1997), so $\mathrm{BQP} \subseteq \mathrm{PSPACE}$. A polynomial-space machine can try every witness in turn, reusing its space, so $\mathrm{NP} \subseteq \mathrm{PSPACE}$, and the same holds level by level for PH.

None of these containments is known to be strict. Since $\mathrm{BQP} \subseteq \mathrm{PSPACE}$ and $\mathrm{P} \subseteq \mathrm{BPP}$, a proof of $\mathrm{BQP} \neq \mathrm{BPP}$ would also prove $\mathrm{P} \neq \mathrm{PSPACE}$, for which no formal proof is known. BQP is not known to contain NP, nor NP to contain BQP; factoring is in BQP by Shor's algorithm and is not known to be in BPP, and no NP-complete problem is known to be in BQP.

No implication between the two separations is known in either direction. If $\mathrm{P} = \mathrm{NP}$, then the whole polynomial hierarchy collapses to P, and BPP, which lies inside the second level of the hierarchy (Sipser, 1983; Lautemann, 1983), equals P; $\mathrm{BQP} \neq \mathrm{BPP}$ would then say that quantum computation reaches beyond the collapsed hierarchy. If $\mathrm{P} \neq \mathrm{NP}$, nothing is known to follow about BQP and BPP. The two questions, P versus NP and BQP versus BPP, concern different ways of branching, and each question is studied on its own terms.

2.8. Summary

P and NP differ in that NP accepts when some witness exists and P must decide with one deterministic path. BPP and BQP differ in that BPP combines its paths by counting nonnegative probabilities and BQP combines them by adding complex amplitudes, which can cancel. In each pair the time bound and the standard of a correct answer are the same. Circuit lower bounds give one route to each separation. Chapter 3 sets out the evidence for each.

Chapter 3. The Evidence

ABSTRACT

Each separation is supported by evidence from several separate lines. For P ≠ NP the evidence is the number and variety of NP-complete problems on which every algorithm found so far takes exponential time, the consequences that P = NP would have, and the barrier theorems. For BQP ≠ BPP it is the quantum algorithms with large speedups, chief among them Shor's algorithm for factoring, the hardness results for sampling problems, the oracle separations, and experiments on quantum devices. None of this evidence is a proof, and the chapter states what each line shows.

3.1. The Evidence for P ≠ NP

The NP-complete problems. Since Cook, Levin and Karp (1972), thousands of problems from logic, graph theory, scheduling, number theory, biology and physics have been shown NP-complete. A polynomial-time algorithm for any one of them would give one for all. Each was studied in its own field, often for decades, by people who wanted a fast algorithm for that problem in particular, and for none of them has one been found. The best known algorithms for satisfiability take time exponential in the number of variables, with a base strictly below 2 for restricted forms of the problem.

The consequences of P = NP. If P = NP, the whole polynomial hierarchy collapses to P, finding a proof of a statement of mathematics of length n becomes as easy as checking one, and no one-way function exists, so the foundations of public-key cryptography fail. The reverse does not hold: P ≠ NP is necessary for one-way functions to exist and is not known to be sufficient, since cryptography needs problems that are hard for most instances, not only for some. The consequences of P = NP are evidence of a different kind from the failed searches: they show that P = NP would make a large set of problems, studied independently, easy at once.

The barriers. The barrier theorems of Chapter 4 are not evidence that P $\neq$ NP is true; they apply equally to proofs of P = NP. They are evidence about the proofs: the techniques that settle other separations do not settle this one.

3.2. The Evidence for BQP $\neq$ BPP: Algorithms

Shor's algorithm. Shor (1994) gave a polynomial-time quantum algorithm for factoring integers and for computing discrete logarithms. The best known classical algorithm for factoring, the number field sieve, takes time about $\exp(c\ n^{1/3}\ (\log\ n)^{2/3})$ on an n-bit number. The quantum algorithm reduces factoring to finding the period of a modular exponential function, and finds the period by a quantum Fourier transform, after which the amplitudes of the multiples of the inverse period reinforce and the others cancel. The algorithm is correct and runs in polynomial time; that factoring is not in BPP is unproved.

Grover's algorithm. Grover (1996) showed that a quantum computer finds a marked item among N with about $\sqrt{N}$ queries, where a classical algorithm needs about N. Bennett, Bernstein, Brassard and Vazirani (1997) showed that no quantum algorithm does better in the query model, so for unstructured search the quantum speedup is exactly quadratic. Exponential speedups come from problems with structure that interference can use, such as the periodicity in Shor's algorithm.

Simulation. Feynman (1982) observed that simulating n interacting quantum particles on a classical computer appears to need resources exponential in n, and proposed a computer built of quantum parts. Lloyd (1996) showed that a quantum computer simulates the evolution of local quantum systems in time polynomial in the size of the system and the time simulated. The best known classical methods for general systems take exponential time.

3.3. The Evidence for BQP $\neq$ BPP: Sampling and Experiments

Hardness of sampling. Aaronson and Arkhipov (2011) showed that if a classical machine could sample exactly from the output distribution of a network of beam splitters fed with single photons, the polynomial hierarchy would collapse to its third level; for approximate sampling the same conclu-

sion rests on two conjectures they state, on permanents of Gaussian matrices and on their anticoncentration. Bremner, Jozsa and Shepherd (2011) proved the corresponding result for exact sampling from a class of commuting quantum circuits. The results of Aaronson and Arkhipov and of Bremner, Jozsa and Shepherd are results about sampling tasks. The separation of the decision classes BQP and BPP does not follow from them directly.

Experiments. In 2019 a group at Google reported a sampling task performed on a 53-qubit superconducting processor in about 200 seconds, which they estimated would take a classical supercomputer about ten thousand years (Arute and others, 2019). Photonic experiments on Gaussian boson sampling followed (Zhong and others, 2020; Madsen and others, 2022). Improved classical simulation methods have since narrowed some of the reported gaps. An experiment measures a finite device on finite inputs, and BQP $\neq$ BPP is a statement about growth with the input, so no experiment can prove or refute it; what the experiments show is how costly classical simulation of these devices has become.

3.4. The Evidence for BQP $\neq$ BPP: Oracle and Conditional Separations

Oracle separations. Relative to an oracle, a machine may ask questions of a fixed language and receive answers in one step. Bernstein and Vazirani (1997) gave an oracle relative to which BQP is not contained in BPP, and Simon (1997) gave an oracle problem with an exponential quantum speedup: determine whether a function on n-bit strings is one-to-one or two-to-one with a hidden period; a quantum algorithm does this with polynomially many queries and any classical algorithm needs about $2^{n/2}$. Raz and Tal (2019) proved that relative to an oracle, BQP is not contained in the whole polynomial hierarchy, using the forrelation problem of Aaronson (2010) and Aaronson and Ambainis (2015). Oracle results do not transfer to the unrelativized classes, as Chapter 4 makes precise; they show that the quantum advantage, where an oracle supplies the structure, can be very large.

Conditional separations. If factoring is not in BPP, Shor's algorithm separates BQP from BPP. The same holds for discrete logarithms. Each conditional separation is as strong as its hypothesis, and the hypotheses are among the most widely relied on in cryptography.

3.5. What the Evidence Shows

For each separation the lines of evidence are separate: no line presupposes another, and each is sound within its own domain. None of the lines of evidence is a formal derivation of the separation. For P ≠ NP the evidence is the convergence of independent searches on the same failure, and the reach of what P = NP would imply. For BQP ≠ BPP it is the convergence of algorithms, oracle separations, conditional separations and experiments on the same advantage.

The evidence bears on whether the separations are true. The question here is a different one, what every proof of the separations must contain, and the evidence enters that question only as background. The ground for that question is the barrier theorems, which Chapter 4 states.

3.6. Summary

P ≠ NP is supported by the failure of independent searches across thousands of NP-complete problems and by the consequences P = NP would have. BQP ≠ BPP is supported by Shor's algorithm, the quantum simulation algorithms, the hardness of sampling, oracle and conditional separations, and experiments. The evidence for each is convergent and is not a proof. Chapter 4 states the theorems that show which proofs are not available.

Chapter 4. The Three Barriers

ABSTRACT

Three theorems show that whole families of proof techniques cannot settle P versus NP or BQP versus BPP. The relativization barrier rests on oracles relative to which each pair of classes is equal and oracles relative to which it differs. The natural proofs barrier shows that, if strong pseudorandom function generators exist, no proof that rests on a property of truth tables that is cheap to test and held by many functions shows that any function needs superpolynomial circuits. The algebrization barrier rests on oracles paired with low-degree polynomial extensions, on both sides of each question. This chapter states each barrier as a theorem for both separations, says which families of techniques it covers, and states what a proof must do to avoid all three. Appendix A gives proof sketches.

4.1. Oracles

Definition 4.1 (Oracle machine). An oracle machine is a machine with a query tape and a query state. When it enters the query state with a string q on the query tape, it receives in one step the answer to whether q is in a fixed language O, the oracle. The machine M with access to O is written M^O. A quantum oracle machine may query the oracle on a superposition of strings, receiving the answers in superposition.

Definition 4.2 (Relativized classes; relativizing technique). For an oracle O, P^O, NP^O, BPP^O and BQP^O are the classes defined as in Chapter 2 by machines with access to O. A proof technique relativizes when its argument goes through unchanged if every machine in it is given access to the same arbitrary oracle: a relativizing proof of a statement about classes proves the same statement relative to every oracle.

The oracle answers for free and shows nothing of how it reaches its answers. A relativizing argument therefore uses the oracle's answers as values and nothing more. Diagonalization and simulation relativize: a machine with an oracle can still be enumerated, its steps counted, and its runs simulated by another machine with the same oracle. The time hierarchy theorem (Hartmanis and Stearns, 1965), the space hierarchy theorem (Stearns, Hartmanis and Lewis, 1965), the undecidability of the halting problem and Ladner's theorem (Ladner, 1975) all have relativizing proofs.

4.2. The Relativization Barrier

Theorem 4.3 (Baker, Gill and Solovay, 1975). There are oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$ and $\mathrm{P}^B \neq \mathrm{NP}^B$.

Theorem 4.4 (Relativization for BQP and BPP). There are oracles A and B such that $\mathrm{BQP}^A = \mathrm{BPP}^A$ and $\mathrm{BQP}^B \not\subseteq \mathrm{BPP}^B$. For A any language complete for polynomial space will do; for B, the oracles of Bernstein and Vazirani (1997) and of Simon (1997).

In both theorems the first oracle is the same. Relative to a language A complete for PSPACE, every class between P and PSPACE collapses: a deterministic polynomial-time machine with access to A answers any question of polynomial space in one query, and every machine of the classes above, with access to A, can be simulated in polynomial space, so $\mathrm{P}^A = \mathrm{NP}^A = \mathrm{BPP}^A = \mathrm{BQP}^A = \mathrm{PSPACE}$. The second oracle differs. For P and NP it is built stage by stage, so that a nondeterministic machine can guess a string of each chosen length and check it with one query, while each deterministic machine in turn, having time for too few queries, errs on some length. For BQP and BPP it encodes a function with a hidden structure that a quantum algorithm finds by interference among superposed queries and that every classical algorithm needs superpolynomially many queries to find, exponentially many for Simon's oracle.

Corollary 4.5 (The relativization barrier). No relativizing technique proves $\mathrm{P} \neq \mathrm{NP}$ or $\mathrm{P} = \mathrm{NP}$, and none proves $\mathrm{BQP} \neq \mathrm{BPP}$ or $\mathrm{BQP} = \mathrm{BPP}$.

Proof. A relativizing proof of $\mathrm{P} \neq \mathrm{NP}$ would prove $\mathrm{P}^{O} \neq \mathrm{NP}^{O}$ for every oracle O, and the oracle A of Theorem 4.3 is one for which this fails; a relativizing proof of $\mathrm{P} = \mathrm{NP}$ would fail for B. The same argument with Theorem 4.4 covers BQP and BPP. ■

4.3. The Natural Proofs Barrier

Circuit lower bounds do not relativize in general, since they examine circuits gate by gate. Through the 1980s they gave strong results for restricted circuits. Razborov (1985) proved that the clique function needs monotone circuits of superpolynomial size. Furst, Saxe and Sipser (1984) and Ajtai (1983) proved that parity needs superpolynomial size in constant depth, and Håstad (1987) made the bound exponential. Razborov (1987) and Smolensky (1987) proved lower bounds for constant-depth circuits with counting gates modulo a prime. None of these methods has been extended to general circuits, and Razborov and Rudich showed why.

Definition 4.6 (Natural proof). A proof that a Boolean function f is not computed by polynomial-size circuits is natural when it shows that f has a property C_n of functions on n variables satisfying three conditions. Constructivity: given the truth table of a function on n variables, a string of length 2^n, whether the function has C_n can be decided in time polynomial in 2^n. Largeness: at least a $2^{-\mathrm{O}(n)}$ fraction of all functions on n variables have C_n. Usefulness: no function computed by polynomial-size circuits has C_n, for all large n.

Theorem 4.7 (Razborov and Rudich, 1997). If there are pseudorandom function generators secure against circuits of size $2^{n^{\varepsilon}}$ for some $\varepsilon > 0$, then no natural proof shows that any Boolean function is not computed by polynomial-size circuits.

The reason is short. A natural property, tested on a whole truth table, tells truly random functions apart from the functions that a small circuit computes from a random seed. Usefulness says that the functions a small circuit computes from a random seed never have the property, largeness says that truly random functions often have the property, and constructivity says that the test on the truth table is cheap. A cheap test of the kind that a natural property gives breaks a strong pseudorandom function generator. Generators of this strength follow from one-way functions of the same strength, and their existence is a standard assumption of cryptography.

Corollary 4.8 (The natural proofs barrier). Under the assumption of Theorem 4.7, no natural proof shows that a problem in NP needs superpolynomial circuits, and none shows that a problem in BQP does. So the circuit route to P $\neq$ NP, and the circuit route to BQP $\neq$ BPP, cannot be completed by a natural proof.

The constant-depth lower bounds named above are natural in this sense, and so are the methods behind them: random restrictions and the method of Razborov and Smolensky. The constructivity and largeness of the natural methods, which made those methods powerful against restricted circuits, are what the theorem of Razborov and Rudich turns against those methods for general circuits.

4.4. The Algebrization Barrier

Arithmetization crossed the relativization barrier. It replaces a Boolean formula by a polynomial over a finite field built from the formula's gates, which agrees with the formula on Boolean inputs and can be evaluated at other points, so that a verifier can check claims about the formula by checking the polynomial at random points. IP is the class of problems for which a probabilistic polynomial-time verifier, exchanging messages with an all-powerful prover, accepts every true instance and rejects every false one with high probability; MIP is the same with several provers who cannot communicate with each other. The proofs that IP = PSPACE (Lund, Fortnow, Karloff and Nisan,

1992; Shamir, 1992) and MIP = NEXP (Babai, Fortnow and Lund, 1991) use it, and neither result holds relative to every oracle. Aaronson and Wigderson defined the property that these proofs still have.

Definition 4.9 (Low-degree extension; algebrizing proof). Let O be an oracle, read on each input length n as a Boolean function on $\{0,1\}^n$. A low-degree extension of O is a family of polynomials $\tilde{O}_n$ over finite fields such that $\tilde{O}_n$ agrees with O on every input in $\{0,1\}^n$ and has low degree in each variable. A proof of an inclusion $C \subseteq D$ algebrizes when it also shows $C^O \subseteq D^{\tilde{O}}$ for every oracle O and every low-degree extension $\tilde{O}$ of O. A proof of a non-inclusion $C \not\subseteq D$ algebrizes when it also shows $C^{\tilde{O}} \not\subseteq D^O$ for every such pair. In both cases the extension goes to the class that must be shown the stronger (Aaronson and Wigderson, 2009).

Every relativizing proof algebrizes, since a machine with access to $\tilde{O}$ can read O from it on Boolean points. Aaronson and Wigderson showed that the proofs of IP = PSPACE and of MIP = NEXP algebrize.

Theorem 4.10 (Aaronson and Wigderson, 2009). (i) There are an oracle A and a low-degree extension $\tilde{A}$ of it such that $\mathrm{NP}^{\tilde{A}} \subseteq \mathrm{P}^A$ and $\mathrm{BQP}^{\tilde{A}} \subseteq \mathrm{BPP}^A$. (ii) There are an oracle B and a low-degree extension $\tilde{B}$ of it such that $\mathrm{NP}^B \not\subseteq \mathrm{P}^{\tilde{B}}$, and an oracle B' and a low-degree extension $\tilde{B}'$ of it such that $\mathrm{BQP}^{B'} \not\subseteq \mathrm{BPP}^{\tilde{B}'}$.

For part (i), take A complete for PSPACE and $\tilde{A}$ its multilinear extension, which can itself be computed in polynomial space (Babai, Fortnow and Lund, 1991); this is their Theorem 5.1. A nondeterministic or quantum polynomial-time machine with access to $\tilde{A}$ can then be simulated in polynomial space, and polynomial space is contained in P^A, which is contained in BPP^A. Part (ii) for P and NP follows from their separation of NP^B from $\mathrm{BPP}^{\tilde{B}}$, since $\mathrm{P}^{\tilde{B}} \subseteq \mathrm{BPP}^{\tilde{B}}$. Part (ii) for BQP and BPP is item (v) of their Theorem 5.11, which carries the exponential separation, due to Raz (1999), of quantum from randomized communication complexity into the setting of oracles and extensions.

Corollary 4.11 (The algebrization barrier). No algebrizing technique proves $\mathrm{P} \neq \mathrm{NP}$ or $\mathrm{P} = \mathrm{NP}$, and none proves $\mathrm{BQP} \neq \mathrm{BPP}$ or $\mathrm{BQP} = \mathrm{BPP}$.

Proof. An algebrizing proof of $\mathrm{NP} \not\subseteq \mathrm{P}$ would show $\mathrm{NP}^{\tilde{O}} \not\subseteq \mathrm{P}^{O}$ for every pair, and Theorem 4.10(i) gives a pair for which this fails. An algebrizing proof of $\mathrm{NP} \subseteq \mathrm{P}$ would show $\mathrm{NP}^{O} \subseteq \mathrm{P}^{\tilde{O}}$ for every pair, and Theorem 4.10(ii) gives a pair for which this fails. The same argument covers BQP and BPP. ■

4.5. The Barriers Together

The three barriers were found in different decades by different people studying different methods. Relativization (1975) came from oracle constructions, natural proofs (1997) from the study of circuit lower bounds, and algebrization (2009) from the study of interactive proofs and arithmetization. None was built to complement another.

The algebrizing family contains the relativizing family, since every relativizing technique algebrizes, so the algebrization barrier covers every technique that relativizes or algebrizes, without any assumption. The natural proofs barrier covers techniques that may do neither, and it rests on the existence of strong pseudorandom function generators. Between them the three families cover diagonalization and simulation, the combinatorial lower bounds against restricted circuits, and arithmetization with the interactive proofs built on it.

Family	What its arguments use	Blocked for P vs NP by	Blocked for BQP vs BPP by
Relativizing	oracle answers, as values	Baker, Gill and Solovay (Theorem 4.3)	a PSPACE-complete oracle, with Bernstein and Vazirani and Simon (Theorem 4.4)
Natural	a property of truth tables	Razborov and Rudich (Theorem 4.7), with P $\subseteq$ P/poly	Razborov and Rudich (Theorem 4.7), with BPP $\subseteq$ P/poly
Algebrizing	values of a low-degree extension	Aaronson and Wigderson, their Theorems 5.1 and 5.3 (Theorem 4.10)	Aaronson and Wigderson, their Theorem 5.1 and item (v) of their Theorem 5.11 (Theorem 4.10)

For both separations, the barriers are theorems about the two classes that the separation compares. The oracles and extensions that block P versus NP are built for P and NP; those that block BQP versus BPP are built for BQP and BPP, the separating ones from quantum query and communication results that have no counterpart in the classical case. The two sets of barrier results are independent of each other.

4.6. What a Proof Must Avoid

A proof of either separation must avoid all three families at once. It must use some property of computation that fails relative to some oracle, so that it does not relativize. If it goes through a circuit lower bound, the property it uses to certify hardness must be costly to test or rare among functions, so that it is not natural. And it must use some property that fails when an oracle is replaced by its low-degree extension, so that it does not algebrize.

Each of the three demands on a proof concerns what the proof uses about the machines that the proof reasons about. A relativizing argument sees only the answers of an oracle. A natural proof sees only a property of a truth table. An algebrizing argument sees only the values of a polynomial extension. A proof that avoids all three must use something about the machines that none of these shows. Chapter 5 states what that is.

4.7. Summary

For both separations, three theorems block three families of techniques. Oracles on both sides block relativizing techniques. Under the assumption of strong pseudorandom generators, natural proofs cannot give the circuit lower bounds either separation could be reached through. Oracles paired with low-degree extensions, on both sides, block algebrizing techniques. The barrier results for BQP and BPP are proved for those classes, independently of those for P and NP.

Chapter 5. What the Barriers Share

ABSTRACT

The three barrier families were found separately, and they have one thing in common. Each reasons from what a computation produces: the answers of an oracle, the truth table of a function, the values of a polynomial extension. None reasons from what a machine is, its program or its gates, to what it can do. Both separations assert a relation of exactly that kind: that what the machines of one model are lets them do something in polynomial time that the machines of the other model, being what they are, cannot. This chapter states the observation, and states what Part II must add to make it exact.

5.1. What Each Family Uses

A relativizing argument uses the answers an oracle gives. Relative to a PSPACE-complete oracle every class between P and PSPACE becomes equal, because the oracle answers in one step what a polynomial-space machine would compute. A relativizing argument cannot tell whether a machine found an answer itself or received it from the oracle, because its reasoning goes through the same way whichever oracle is attached.

A natural proof uses a property of a function's truth table. A function computed by a small circuit from a short random seed, the output of a pseudorandom generator, has a truth table that every efficient test finds indistinguishable from a random one. A natural proof cannot tell the pseudorandom truth table from a random one, because a natural proof tests the truth table and not the circuit that produced the truth table.

An algebrizing argument uses the values of a polynomial extension of an oracle. The extension says more about the oracle than its Boolean answers do, and an algebrizing argument still uses the extension only as values returned to queries. Relative to suitable extensions on both sides of each question, the same argument would reach opposite conclusions.

In each case the family uses what a computation produces, at some level of detail, and the barrier exhibits two situations in which those products look the same and the answer to the question differs.

5.2. What the Separations Assert

P ≠ NP asserts that some language with polynomial-time verifiable witnesses is decided by no deterministic polynomial-time machine. A proof must show, for some such language, that every deterministic polynomial-time machine fails to decide it. The statement that every deterministic polynomial-time machine fails to decide the language is a statement about every member of a class of machines, made because of what those machines are: deterministic, with polynomially many steps.

BQP ≠ BPP asserts that some language is decided by a uniform family of polynomial-size quantum circuits and by no polynomial-time probabilistic machine. A proof must show both: that the quantum circuits, being what they are, decide the language, and that the probabilistic machines, being what they are, do not.

In both, the claim relates what the machines of a model are to what they can do. The time bound and the standard of a correct answer are the same on the two sides, and the claim is that the way the machines branch makes the difference.

5.3. The Mismatch

Put the two observations together. Each barrier family reasons from what a computation produces, and each separation is a claim about what machines can do because of what they are. A proof of either separation must, somewhere, reason from what the machines of a model are to what they can or cannot do, and the three families, by the barrier theorems, do not reach either separation.

The observation suggests that the three barriers are three instances of one limit: that every technique that reasons only from what computations produce falls short of both separations, and that a proof of either must contain reasoning of another kind. As it stands, the observation is a description, and it does not yet support a conclusion. Three things are needed to make it exact.

First, a definition of the kinds of reasoning a proof can contain, precise enough to sort every step of a formal proof. Second, a test that fixes the boundary between the first kind and the third, and an assignment of each barrier family to the kinds its techniques use, checked against the test. Third, a lemma that turns the failure of relativizing techniques into a statement about every proof. The FIS book supplies the definitions and the sector-elimination lemma. Chapter 6 imports them and fixes the test, Chapter 7 states what they do not decide, and Chapter 8 assigns the families, proves the covering lemma and builds the template.

5.4. Summary

Relativizing, natural and algebrizing techniques each reason from what a computation produces, and each barrier exhibits two cases that look the same to the family and differ in the answer. Both separations assert that what a model's machines are changes what they can do. The observation that a proof must reason from what the machines are to what they can do is made exact in Part II, through the three kinds of step and the test between them, the covering lemma, and the sector-elimination lemma.

PART II

The Template

Chapter 6. The Kinds of Step

ABSTRACT

This chapter imports from the FIS book the definitions on which the template rests, and fixes the test that draws the boundary between the first kind of step and the third. Every step of a formal proof is of one or more of three kinds. The engagement of a proof is the set of kinds it uses, and the required engagement of a proposition is the set of kinds that every formal, sound proof of it uses, among the proofs that do not assume it openly, with the third kind counted only when it derives what the proposition's own processes do or cannot do, as the proposition states it. Propositions are sorted by required engagement into Type S, Type P and third-kind propositions. The sector-elimination lemma turns the failure of a collection of technique families into a statement about every proof. Each definition is stated in full with its number in the FIS book.

6.1. Three Kinds of Step

Definition 6.1 (The three kinds of step; engagement; FIS Definition 7.1). Every step of a formal proof, and every premise it cites, is of one or more of three kinds. A step is of the first kind (e_1) when its verdict does not depend on the order of its terms: checking identity, substituting equals for equals, applying a definition, counting, enumerating, and building a definite object from definite inputs. A step is of the second kind (e_2) when it relates two terms in roles that cannot be exchanged without changing the step: case analysis whose cases are handled by different arguments, conditional branching, an oracle query and its response, a challenge and its answer. A step is of the third kind (e_3) when it derives what a process does from what the process is, or derives a limit on what the members of a class of processes can do from what those members are. The engagement $\mathrm{Eng}(\pi)$ of a proof π is the set of kinds used by its steps and by the premises it cites. A proof is said to use, or to contain, a step of a kind when that kind is in its engagement, whether from one of its own steps or from a premise it cites.

The relation between what a process is and what it does is its structure/function relation. A step of the third kind takes that relation as the object of its reasoning. Every formal proof is itself a process whose form fixes its output; that fact holds of every proof and is not, by itself, a step of the third kind.

Three points of reading follow the FIS book. Engagement belongs to a proof, not to a proposition: two proofs of one proposition can use different kinds. Every proof uses the first kind, since identity and substitution are present in every derivation. And the kinds classify the method, not the topic: a proof about machines can use only steps of the first kind, as the proof of the halting theorem does. A premise supplies the kinds its content carries: a cited result that asserts what some process does because of what it is supplies a step of the third kind to any proof that uses it.

6.2. The Tests

Definition 6.2 (Engagement template; FIS Definition 7.3). A proof technique uses the first kind when it employs steps whose verdict is the same whatever the order of their terms: identity checks, substitution, definition-application, counting, enumeration, and the construction of definite objects from definite inputs. Every proof uses e_1. It uses the second kind when it employs steps in which two terms play roles that cannot be exchanged: case analysis whose cases are handled by different arguments, conditional branching, oracle access in which the query and the answer play different parts, interactive or adversarial protocols. Two tests fix the boundary between the first and second kinds. A branch whose cases are handled by one and the same argument applied to different values (a uniform map, as in a diagonal construction, which changes each value by one rule) is of the first kind. An interface whose two roles are present but whose reasoning would go through unchanged with the roles exchanged is of the first kind. The relation of premise to conclusion, present in every proof, is not by itself a step of the second kind. It uses the third kind when a step of its reasoning derives what a process does from what the process is, or a limit on what a class of processes can do from what its members are, with that derivation as the object of the step. A definition that refers to itself only through an encoded object (a diagonal or fixed-point construction) is of the first kind.

In the FIS book I analyze three model cases with the tests set out in this section. Diagonalization uses steps of the first kind only: it builds an object that differs from each member of a list at that member's own index, by one rule applied alike to every member (FIS Proposition 7.5). Interactive proofs use the first two kinds: within each message the parties compute by identity and substitution, and between messages the prover proposes and the verifier tests, in roles that cannot be exchanged (FIS Proposition 7.6). A proof that a given program sorts every list uses all three: its central step derives what one pass of the loop does to every array from the text of the loop body, which is a derivation of what a process does from what it is (FIS Section 7.8).

The definitions leave one line to be drawn: where a step that runs a process, counts it or applies its definition ends, and a step that derives what the process does from what it is begins. The boundary test, stated next, draws that line.

Definition 6.2a (The boundary test). A computation, in the defined sense, is a sequence of configurations, each obtained from the one before by one application of the process's step rule, together with the resources the sequence uses (its number of steps, the length of its configurations, its random choices or quantum bits). A step uses a process only as a computation when it runs the process, directly or by a universal machine; enumerates processes; counts their steps, configurations or number; builds a definite object from a description of the process; branches on what the process outputs; or applies a definition. Such a step is of the first kind, or of the second by the tests of Definition 6.2. A step is of the third kind when its object is the inner makeup of a process (the particular form of its step rule, such as the fact that each step of a Turing machine changes only a bounded window of its tape; the kinds of its gates, such as the fact that a gate computes parity; or the body of its loop) and what that makeup makes the process do, or what the makeup of the members of a class limits them to doing. A cited premise carries the kinds of the steps of its proof; a premise with no proof except from itself, a primitive axiom or a definition, carries the first kind; and either kind of premise also carries the third kind when its content asserts what some process does because of what it is (Section 6.1).

Two cases need saying, because each keeps a calibration below.

The first case is the universal machine. Its existence is Turing's theorem (Turing, 1936): there is a machine that, given the description of any machine and an input, produces that machine's output. The proof builds the universal machine to store the other machine's transition table as data and, at each step, look up the entry for the current state and symbol and apply it. Each cycle of the universal machine is one application of the simulated machine's transition function, so its run is the simulated machine's computation in the defined sense, carried out step for step. The proof that the universal machine works checks a construction whose steps are, one for one, the steps of a definition,

and that proof is of the first kind. Running a machine by a universal machine is therefore a step of the first kind. The proof that a given program sorts every list is different: being sorted is not the rule the loop applies, so the proof must derive, from the body of the loop, what every pass does, and that step is of the third kind.

The second case is construction. Building an object from a process's description is of the first kind, and so is any fact about the object that follows from the building rule alone, such as its size. A claim that the object captures what the process does, derived from the inner form of the process's step rule, is of the third kind. The construction in the theorem of Cook and Levin is of the first kind; the claim that the formula it builds is satisfiable exactly when the machine accepts is of the third, since it rests on each step's changing only a bounded window of the tape.

The tests at work. Turing's proof of the halting theorem shows the tests applied step by step. Turing's proof supposes a decider, which the proof later discharges, so the decider is a supposition and not a premise. It builds a program that, on its own description, runs the decider and does the opposite of what the decider predicts: a definite object built from definite inputs by one rule applied to every candidate decider, a step of the first kind by the uniform-map test. Turing's proof then reasons in two cases, that the decider predicts the program halts and that the decider predicts the program does not halt, and each case is closed by the same mirrored argument, so the interface test counts the branch as of the first kind. The decider and the program are used only as computations: run, and their outputs compared. The engagement of the proof is $\{e_1\}$. The proof that a given program sorts every list takes, at its central step, the body of the loop as its object and derives from it what every pass does to every array, a step of the third kind. The test classifies the other cases this book uses as follows.

Proof or step	What the step uses	Kind
Halting theorem: the diagonal program	Runs the supposed decider and reverses its answer, by one rule for every candidate	First
Time hierarchy theorem: the diagonal machine	Runs each smaller machine by a universal machine, counts steps, reverses the answer	First
A given program sorts every list	Derives from the loop body what every pass does to every array	Third
The SAT verifier is correct	Evaluating a formula on an assignment is the definition of the assignment satisfying it; its time is a count	First
Cook and Levin: the formula is satisfiable exactly when the machine accepts	Derives from the bounded window of each step that the accepting computations are exactly the locally consistent tableaux	Third
Interactive proofs decide polynomial space: completeness of quantified Boolean formulas	Rests on the same bounded-window property of polynomial-space machines	Third, by the premise it cites
Smolensky: constant-depth circuits with parity gates	Derives from the kinds of the gates that every such circuit agrees with a low-degree polynomial on most inputs	Third
Williams: the Beigel and Tarui representation	Derives from the gates of a constant-depth circuit with counting gates a representation that a faster algorithm uses	Third
A natural proof: usefulness of its property	Derives from what small circuits are that none of them has the property	Third, by the premise it cites
Shannon's count: most functions need large circuits	Counts circuits of each size and functions	First

6.3. Required Engagement

> **Definition 6.3 (Required engagement; FIS Definition 7.2).** Let *Prop* be a proposition. Consider every proof π of *Prop* that is formal and sound, and that assumes no aspect of *Prop* openly (Definition 7.2 below). An aspect of *Prop* is a claim *Prop* asserts, whole or in part; an argument assumes a claim openly when one premise it does not establish, by itself, implies the claim. Chapter 7 states both definitions in full. The required engagement $\mathrm{Req}(Prop)$ is the set of kinds of step that every such proof uses: the intersection of $\mathrm{Eng}_{Prop}(\pi)$ over all of them. Here $\mathrm{Eng}_{Prop}(\pi)$ is $\mathrm{Eng}(\pi)$ with one restriction and one addition, both on the third kind. The restriction: a step of the third kind is counted only when the process it reasons about, and what it derives that process does or cannot do, are part of what *Prop* states. The addition: e_3 is also counted when one of the conditions π must meet to count as a formal proof is that π itself, by what it is, does something that *Prop* states it does, even if no step of π reasons about that.

The exclusion and the restriction have one reason: $\mathrm{Req}(Prop)$ measures what reaching *Prop* demands. Any consistent proposition is derivable in the system that adopts it as an axiom, by the one-line proof "*Prop*, by axiom *Prop*"; that proof starts at *Prop* and says nothing about what reaching it requires, and it is left out. A step of the third kind about a process that *Prop* is not about is a detour that *Prop* does not demand, and it is not counted.

$\mathrm{Req}(Prop)$ takes its value among the subsets of $\{e_1, e_2, e_3\}$, and the value is fixed by three questions, one for each kind: is there a proof of *Prop* in the class that does not use this kind? The intersection is defined only when *Prop* has a proof in the class. For a proposition classified before any proof of it is examined, the FIS book uses an operative form, which I use throughout this book: a proposition requires e_3 when it is shown that every proof of it in the class, should one exist, counts e_3 in $\mathrm{Eng}_{Prop}(\pi)$ (FIS Sections 7.3 and 8.1). A requirement stated in this operative form is a statement about every proof that might exist, and such a requirement is established by an argument, not by a survey.

6.4. Engagement Bounds

Definition 6.4 (Engagement bound; FIS Definition 7.4). A proof technique class Π is E-bounded, where $E \subseteq \{e_1, e_2, e_3\}$, when every proof π in Π has $\mathrm{Eng}(\pi) \subseteq E$.

An engagement bound says that no proof in the class uses a kind outside E. It does not say that every proof in the class uses every kind in E. Bounds pass to subclasses: if Π_2 is E-bounded, so is every class contained in it.

6.5. The Typology

Definition 6.5 (Type S, Type P, third-kind proposition; FIS Definitions 8.2 to 8.4). A proposition *Prop* is Type S when $\mathrm{Req}(Prop)$ is $\{e_1\}$: for each of the second and third kinds, some formal, sound proof of *Prop* in the class does without it. *Prop* is Type P when $\mathrm{Req}(Prop)$ is $\{e_1, e_2\}$: every such proof uses a step of the second kind, and some such proof counts no step of the third kind. *Prop* is a third-kind proposition when e_3 is in $\mathrm{Req}(Prop)$: every formal, sound proof of *Prop* that assumes no aspect of *Prop* openly uses a step of the third kind that derives, from what a process is, what it does or cannot do, where that doing is part of what *Prop* states, or counts e_3 by the addition of Definition 6.3.

The three classes are exclusive, and they cover every proposition for which $\mathrm{Req}(Prop)$ is defined (FIS Proposition 8.5). They are not ranked by difficulty: Fermat's last theorem took centuries, and the Wiles proof uses no step that derives what a process does from what it is. The three classes record what kind of reasoning a proof must contain.

The FIS book calibrates the typology with three examples. The undecidability of the halting problem is Type S: Turing's proof builds a diagonal program by one rule applied to every candidate decider, and its steps are of the first kind. Gödel's first incompleteness theorem is Type S: its proof encodes syntax, applies a fixed-point construction, and analyzes the resulting sentence, all by steps of the first kind. The correctness of a given sorting program

has proofs that use a step of the third kind, and such proofs are formal, independent and sound. The first two examples show that the typology does not sort by topic, since both are about machines and proofs; the third shows that needing a step of the third kind is a different thing from assuming part of what is proved.

6.6. The Sector-Elimination Lemma

> **Lemma 6.6 (Sector elimination; FIS Lemma 7.7 and Lemma 12.1).** Let *Prop* be a proposition, and let the proofs considered be the formal, sound proofs of *Prop* that assume no aspect of *Prop* openly. Let Π_1, ..., Π_m be technique classes such that every proof in that class whose engagement lies within $\{e_1, e_2\}$ belongs to some Π_i (the covering condition). If no proof in any Π_i establishes *Prop*, then every proof in the class has e_3 in its engagement: it uses a step of the third kind, or cites a premise that supplies one.

Proof. Suppose a proof π in the class had $\mathrm{Eng}(\pi)$ within $\{e_1, e_2\}$. By the covering condition π belongs to some Π_i. No proof in Π_i establishes *Prop*, and π establishes *Prop*: a contradiction. So the engagement of every proof in the class contains a kind outside $\{e_1, e_2\}$, and the only such kind is the third. ■

The deduction does not use the kinds of step the families Π_i themselves use. The engagement bound of each family is the reason that family fails to reach *Prop*; the conclusion rests on the covering condition and on the failure of each family, and on nothing else.

The lemma concludes something about $\mathrm{Eng}(\pi)$ for every proof in the class. The lemma concludes something about $\mathrm{Req}(Prop)$ only with one more fact: that the step of the third kind each proof then has, in the proof or in a premise it cites, derives what a process Prop is about does or cannot do, as Prop states it. When that is shown, or stated as a premise, e_3 is in $\mathrm{Eng}_{Prop}(\pi)$ for every proof in the class, and *Prop* is a third-kind proposition. The strength of the conclusion follows the strength of the covering condition: where the covering is established, the conclusion is established; where it is stated as a premise, the conclusion holds under that premise.

6.7. Summary

Steps of the first kind compare values without regard to order, steps of the second kind relate two terms in roles that cannot be exchanged, and steps of the third kind derive what a process does from what it is. The required engagement of a proposition is what every formal, sound proof of it that does not assume it openly must use, with the third kind counted only when it derives what the proposition's own processes do or cannot do. The boundary test fixes where the first kind ends and the third begins: a step that uses a process only as a computation is of the first kind, and a step that reasons from its inner makeup to what it does is of the third. A third-kind proposition is one whose required engagement contains the third kind. The sector-elimination lemma turns the failure of technique families that together contain every technique confined to the first two kinds into the statement that every proof uses the third. Chapter 7 states what these definitions do not decide.

Chapter 7. What a Proof Must Do and What It Must Assume

ABSTRACT

This chapter imports from the FIS book the three properties demanded of a proof, the definition of assuming a claim openly, silently, or through what an argument rests on, the inescapably circular propositions, Type T, and the FIS Theorem. It states why the question of Chapter 6, what every proof of a proposition must do, is a different question from what every proof of it must assume, and why a third-kind classification does not apply the FIS Theorem. A proof can derive what the processes its conclusion is about do from what they are, and assume nothing about its conclusion.

7.1. Three Properties of Proof

The FIS book states three properties that every proof is expected to have.

Desideratum F (Formalizability; FIS Section 9.1). The proof is a finite derivation in a stated formal system, each of whose steps yields a definite result and is built from finitely many primitive steps, so that a definite, finitely specified procedure can check every step against the rules.

Desideratum I (Independence; FIS Section 9.2). No step of the proof assumes anything that the proof seeks to prove, openly, silently, or through what it rests on. No premise that the proof does not establish, and no definition that the proof uses, by itself implies the proposition or an aspect of the proposition.

Desideratum S (Soundness, or correctness; FIS Section 9.3). Every step of the proof follows from earlier steps by rules that preserve truth; its premises are true of what they are about; and its last line is the proposition itself, as it is stated.

A proof is valid, in the sense the FIS book uses and I follow in this book, when it is at once formal, independent and sound (FIS Definition 9.2a). A formal derivation of *Prop* is a formal argument every step of which follows by the rules and whose last line is *Prop*; its premises need not be true and it need not be independent.

7.2. Assuming a Claim

> **Definition 7.1 (Aspect of a proposition; FIS Definition 9.1).** An aspect of a proposition *Prop* is a claim that *Prop* asserts, whole or in part. *Prop* itself is an aspect of *Prop*. When *Prop* asserts several claims together, each of them is an aspect of *Prop*. When *Prop* makes a claim about every member of a class, what *Prop* says of any one member of the class is an aspect of *Prop*. A claim that occurs inside *Prop* without being asserted by it is not an aspect of *Prop*.

Definition 7.2 (Assuming a claim, openly, silently, or through what an argument rests on; FIS Definition 9.2). A premise of an argument is a claim that the argument takes as given and that is still in force at the argument's last line; a supposition that is discharged before the end of the argument is not a premise. An argument assumes a claim openly when one premise of the argument, by itself, implies the claim, and the argument does not establish that premise. "By itself" means with no other premise of the argument, though with the rules of logic and the definitions of the terms the premise uses. An argument assumes a claim silently in two cases. In the first case, the statement that the argument meets one of the conditions that the argument must meet to count as a formal proof, or performs one of the actions that the argument must perform to count as a formal proof, by itself implies the claim. In the second case, the claim is about the argument, and the claim holds of the argument because the argument meets such a condition or performs such an action. A definition the argument uses is tested in the same way as a premise: the argument assumes a claim openly when one definition it uses, by itself, implies the claim, and a creative definition is tested as a premise. An argument also assumes a claim silently when the statement that it meets the condition on its rules that it must meet to count as a sound proof (that every rule it applies takes true lines only to true lines) by itself implies the claim, or the claim is about the rules it applies and holds of them because it meets that condition. An argument assumes a claim through what the argument rests on when three things hold. First, the argument rests on a premise, or applies a rule of inference, that the argument does not establish. Second, that premise or rule has formal derivations that do not assume that premise or rule openly. Third, every one of those derivations assumes the claim, openly, silently, or in this way in turn. A premise or rule with no derivation except from itself (a primitive axiom, or a definition) carries no assumption in this way.

Definition 7.3 (Inescapably circular proposition; FIS Definition 9.3). A proof of *Prop* is circular when it assumes *Prop* or an aspect of *Prop* in one of the forms of Definition 7.2: through a premise, stated or silent, or a definition, that by itself cannot not imply that aspect, or through a premise or rule it rests on that can be derived only by assuming that aspect. *Prop* is inescapably circular when every formal derivation of *Prop* cannot not assume some aspect of *Prop* in one of these forms.

Definition 7.4 (Type T; FIS Definition 9.4). A proposition is Type T when it is a third-kind proposition (Definition 6.5) and inescapably circular (Definition 7.3).

Theorem 7.5 (The FIS Theorem; FIS Theorem 2). Let *Prop* be an inescapably circular proposition. Then no proof of *Prop* is at once formal, independent and sound. In particular this holds for every Type T proposition.

***Proof* (*as in the FIS book*).** A proof of *Prop* that is formal and sound is a formal derivation of *Prop*. Since *Prop* is inescapably circular, it has a premise, stated or silent, that by itself implies an aspect of *Prop*, and so it assumes that aspect. Desideratum I forbids this. ■

7.3. Two Different Questions

The definitions of Chapter 6 say what every proof of a proposition must do. The definitions of this chapter say what every proof of a proposition must assume. The FIS Theorem rests on the second alone: to apply it to a proposition is to name an aspect of the proposition and show, by a formal argument, that every formal derivation of it assumes that aspect. The FIS book calls this demonstration Criterion 5, and states that nothing else applies the FIS Theorem (FIS Section 12.8).

The third-kind question and the circularity question have different answers, and the answer to each can hold without the other. A proposition can need a step of the third kind and be proved validly. A proof that a given program sorts every list derives what the program does from what it is, and its

premises are the text of the program and the rules of the language, neither of which is the claim that the program sorts or any part of it. Whether its steps follow does not depend on the program being correct: a proof attempt for an incorrect program fails by the same rules. Such proofs are formal, independent and sound, and they are written every day. Reasoning about a relation is not assuming it.

Conversely, a proposition can be inescapably circular without asserting anything about what a process does. "Every formal proof is finitely specifiable" is about every formal proof, and every formal proof of it counts as a formal proof only because it is finitely specified; the FIS book proves that no proof of it in any formal system is at once formal and independent (FIS Theorem 2.1).

The FIS book shows one proposition that answers both questions. The Church–Turing thesis is inescapably circular, because a derivation counts as a formal proof only in a system whose derivations a Turing machine can check, and that a Turing machine can do this checking is part of what the thesis asserts; and it is a third-kind proposition, because every proof of it must derive which functions Turing machines compute from what Turing machines are, or take that from a premise. So it is Type T (FIS Section 10.8).

7.4. What a Third-Kind Classification Says

For a separation, a third-kind classification says that every proof of it must contain a step that derives what the machines of one model can do, or what the machines of the other cannot do, from what those machines are. It says nothing about whether such a proof assumes part of the separation. A proof that derives a limit on a class of machines from their definitions, with those definitions as its premises, assumes nothing about the separation.

I therefore make no claim that either separation is inescapably circular, no claim that either is Type T, and no claim that either lacks a proof that is formal, independent and sound. Neither separation is a claim about every member of a class to which every formal proof belongs, which is the first condition of a self-dependent proposition (FIS Definition 10.1). A reader may ask whether P ≠ NP is such a claim, since every formal proof can be checked by a deterministic machine in polynomial time. It is not: P ≠ NP is a claim about the ma-

chines, and a proof is checked by one of them without being one of them; Section 11.2 shows, besides, that the general fact that a machine's table fixes its output is not what P ≠ NP asserts. So the route of the FIS book's self-dependence theorem does not apply to either. Nor do I give in this book a demonstration in any of the other forms the FIS book recognizes: through a definition, through the condition on a proof's rules, through what a proof rests on, or by closing every route class by class (FIS Lemma 9.5). The result here is the third-kind classification, and I state at each point that this classification is the whole of what I claim.

7.5. Summary

A proof is valid when it is formal, independent and sound. A proposition is inescapably circular when every formal derivation of the proposition assumes part of the proposition. The derivation can assume that part through a premise, stated or silent, or a definition, that by itself implies that part; or through a premise or rule that the derivation rests on and that can be derived only by assuming that part. For an inescapably circular proposition, no proof is valid. The third-kind question asks what every proof must do; the circularity question asks what every proof must assume; and only the second applies the FIS Theorem. A proof can derive what the processes its conclusion is about do from what they are, and assume nothing about its conclusion. Chapter 8 gives the template for the first question.

Chapter 8. The Template for a Separation

ABSTRACT

This chapter gives one template for showing that a separation of two complexity classes is a third-kind proposition, and proves the template. A separation says that some language of the stronger class lies outside the weaker one. Every proof of it must show that some machine of the stronger model decides a language and that no machine of the weaker model does, and so every proof derives, from what the machines of each model are, what they can or cannot do. I assign the three barrier families their kinds of step under the boundary test, and I prove the covering lemma: every argument confined to the first two kinds relativizes. The template takes as inputs the relativization barrier for the pair of classes, the covering lemma and a step premise, and gives the third-kind classification by the sector-elimination lemma. I state how the template is checked and what the template does not say.

8.1. Separations of Two Classes

The two separations of this book have the same form. There are two classes, a weaker class D and a stronger class C, with $D \subseteq C$ known: P $\subseteq$ NP, and BPP $\subseteq$ BQP. Both classes are defined by machines of a model of computation, under the same bound on time and the same standard of a correct answer. The separation is the claim $C \not\subseteq D$.

Definition 8.1 (Separation). Let $D \subseteq C$ be complexity classes, each defined by the polynomial-time machines of a model of computation. The separation Sep(C, D) is the proposition that some language in C is not in D. It asserts that there is a language L such that (a) some machine of the model of C, running in polynomial time, decides L to the standard of C; and (b) no machine of the model of D, running in polynomial time, decides L to the standard of D.

P ≠ NP is Sep(NP, P), and BQP ≠ BPP is Sep(BQP, BPP). In both cases D is contained in P/poly: P trivially, and BPP by the theorem of Adleman, Bennett and Gill. So in both cases a language in C that needs superpolynomial circuits lies outside D, and a circuit lower bound is one route to the separation.

8.2. What Every Proof of a Separation Must Show

The FIS book's procedure begins with two filters for the third-kind question (FIS Sections 12.2 and 12.3). The procedure has five criteria in all. Criterion 1 asks whether the proposition asserts that what a process is determines or limits what the process does. Criterion 2 asks whether every proof of the proposition derives, from what the proposition's processes are, what those processes do or cannot do. Criterion 3 asks whether technique families that together contain every proof confined to the first two kinds each fail to establish the proposition, so that every proof needs the third kind. Criterion 4 records the results that bear on the proposition. And Criterion 5 asks whether every formal derivation of the proposition must assume part of the proposition, and Criterion 5 alone applies the FIS Theorem. Criteria 1, 2 and 3 are stated here; Chapter 13 states Criterion 4, and Chapter 7 Criterion 5.

> **Criterion 1 (The structure/function predicate; FIS Section 12.2).** Determine whether the proposition asserts that what some process, object or system is determines, limits or fails to limit what it does.

> **Criterion 2 (The load-bearing test; FIS Section 12.3).** Determine whether every formal proof of *Prop*, in its own steps or in the results it cites, contains a derivation, in one step or in many, of what the processes *Prop* is about do from what they are, or of a limit on what they can do from what they are.

For a separation both filters are passed for the same reason. Sep(C, D) says that the machines of one model, being what they are, decide some language, and that the machines of the other, being what they are, do not; the two classes have the same bound on time, and the claim is that the model makes the difference. So the separation passes Criterion 1.

In the terms of comparison, which the FIS book uses, a separation asserts that context constrains content. Context is what a process is, its structure; content is what it does, its function. A structural equivalence matches the definitions of one pair of terms with another and shows two inclusions: each defining term falls within the definition of the other, and the form of the one claim falls within the form of the other. Sections 9.2 and 11.2 show these inclusions for each separation.

A proof of Sep(C, D) must establish (a) and (b) of Definition 8.1 for some language, itself or by citing results that do, since the separation is their conjunction for some L. A proof of (a) takes the definition of some machine of the model of C and concludes what it does on every input: a derivation of what a process does from what it is. A proof of (b) takes the definition of the class of machines of the model of D and concludes that none of them decides L: a derivation of a limit on what the members of a class can do from what they are. A proof that cites (a) or (b) brings the steps of the cited proof with it. So every proof of the separation derives, from what the machines are, what the machines of the model of C do or what the machines of the model of D cannot do, and the separation passes Criterion 2.

Criterion 2 does not settle what kind of step a proof uses to derive what the machines can or cannot do from what the machines are. A derivation of a limit on a class of machines can be made by steps of the first kind: the time hierarchy theorem shows that no machine running in time n^k decides a certain language, by building a diagonal machine that differs from each such machine on some input, by one rule applied alike to every machine. The diagonal construction of the time hierarchy theorem is a derivation of a limit, made by comparing values. Whether every proof of the separation needs a step of the third kind is the question of Criterion 3.

The boundary test of Section 6.2 settles the kind of step that proves part (a) for the languages this book uses. For SAT, the verifier evaluates the formula on the proposed assignment, and evaluating a formula on an assignment is the definition of the assignment satisfying it; so the proof that the verifier is correct applies a definition, and its running time is a count, both of the first kind. For P $\neq$ NP through SAT, part (a) is therefore proved by steps of the first kind, and the step of the third kind every proof needs falls on part

(b). For BQP ≠ BPP, the proof of part (a) for an algorithm such as Shor's derives the output distribution from the particular gates of the circuit, its quantum Fourier transform among them, and is itself of the third kind by the test; Premise L (Section 8.5) counts a step of the third kind in either part.

Criterion 3 (Sector elimination; FIS Lemma 12.1). Determine whether technique families that together contain every proof confined to the first two kinds of step each fail to establish the proposition. If they do, every proof of it uses a step of the third kind (Lemma 6.6).

8.3. The Engagement Bounds of the Three Families

The FIS book assigns each of the three barrier families an engagement bound, by applying the tests of Definition 6.2 to the operations the technique's arguments use (FIS Section 12.6). With the boundary test of Section 6.2, the relativizing techniques use their oracles only in steps of the first kind, and the natural and algebrizing techniques each take a step of the third kind, in a form their barrier excludes.

Proposition 8.2 (Relativizing techniques; FIS Proposition 12.2). Relativizing techniques are $\{e_1, e_2\}$-bounded, and every step by which a relativizing argument uses its oracle is of the first kind.

A relativizing argument must go through for every oracle, so it uses each oracle answer as a value fed to one argument: it simulates the queried machine, or it diagonalizes by a uniform map that changes each value by one rule. The direction of a query, from the machine to the oracle and back, is part of the way the values arrive, and the reasoning performed on them would go through unchanged with the roles exchanged, which the template counts as the first kind. An argument that used the makeup of a particular oracle, how it reaches its answers, would not go through for every oracle. So the argument takes no step that derives what the oracle does from what the oracle is, or what a machine does through the oracle from what that machine is.

Proposition 8.3 (Natural proof techniques; FIS Proposition 12.3). Natural proof techniques take a step of the third kind, in the proof that their property is useful, and use its result only through a test on truth tables.

A natural proof counts functions and tests a property of truth tables, which are steps of the first kind, and it classifies functions by whether they have the property, which is a step of the second kind. Its usefulness condition (Definition 4.6: no function computed by polynomial-size circuits has the property) is a limit on what every circuit of a class can compute, and its proof takes what the circuits are as its object. In Smolensky's lower bound, the step that every constant-depth circuit with parity gates agrees, on most inputs, with a polynomial of low degree over the two-element field derives, from the kinds of the gates, what the circuits cannot compute. By the boundary test that step is of the third kind, and a natural proof that cites its usefulness lemma carries it. The natural proof then uses the usefulness lemma only through the natural property, which is cheap to test on a truth table and held by many functions; the natural proofs barrier shows that a step of the third kind so used cannot give a superpolynomial circuit lower bound.

Proposition 8.4 (Algebrizing techniques; FIS Proposition 12.4). Algebrizing techniques take a step of the third kind, through the completeness results they cite, and use its result only in the arithmetized form that a low-degree extension preserves.

An algebrizing argument does arithmetic over a finite field and evaluates polynomials, which are steps of the first kind, and it reasons about the relation between an oracle and its low-degree extension, which restricts to the oracle on Boolean points and not the reverse, and about degree bounds, which are steps of the second kind. Its reach beyond relativization comes from what it arithmetizes. In the proof that interactive proofs decide exactly polynomial space, that is the completeness of quantified Boolean formulas for polynomial space, which rests, as the theorem of Cook and Levin does, on each step of a machine's changing only a bounded window of its tape. The bounded-window property is a fact about the inner makeup of a machine's step rule, used

to derive what every machine's accepting computations are, and by the boundary test the bounded-window property is of the third kind. Arora, Impagliazzo and Vazirani (1992) identified the bounded-window property, as local checkability, as the property that non-relativizing techniques use. The algebrizing argument then uses that fact about the step rule only in the form that a low-degree extension preserves. The conclusions of the algebrizing argument hold for every oracle together with every extension, so the argument uses the oracle and the oracle's extension only as values returned to queries, and uses no property of a particular computation that those values do not show.

These assignments are classifications of each family's operations under the tests of Section 6.2, checked operation by operation, and not formal theorems of the kind the barrier results are. A counterexample to Proposition 8.2 would be a relativizing technique whose argument takes a step of the third kind; a counterexample to Proposition 8.3 or 8.4 would be a technique in that family that reaches beyond relativization by steps of the first two kinds alone, which the covering lemma excludes. The template does not use them in its deduction: the sector-elimination lemma rests on the covering lemma and on the relativization barrier, and the assignments say what each of the three barriers shows about the kinds of step.

8.4. The Barrier Results for a Pair of Classes

Definition 8.5 (The three families for a separation). Let the proofs considered be the formal, sound proofs of Sep(C, D) that assume no aspect of it openly. Such a proof is in the relativizing family when it relativizes; in the algebrizing family when it algebrizes; and in the natural family when it establishes the separation by showing, through a natural property (Definition 4.6), that some language in C is not in P/poly.

The barrier results for the pair are three statements.

(R) There is an oracle A such that $C^A \subseteq D^A$. Then no proof in the relativizing family establishes the separation, since such a proof would show $C^O \not\subseteq D^O$ for every oracle O.

(A) There are an oracle A and a low-degree extension $\tilde{A}$ of it such that $C^{\tilde{A}} \subseteq D^A$. Then no proof in the algebrizing family establishes the separation, since such a proof would show $C^{\tilde{O}} \not\subseteq D^O$ for every pair.

(N) There are pseudorandom function generators secure against circuits of size 2^{n^ε} for some $\varepsilon > 0$. Then, by Theorem 4.7, no natural proof shows that any language needs superpolynomial circuits, and no proof in the natural family establishes the separation.

Since every relativizing proof algebrizes, (A) alone excludes both the relativizing and the algebrizing families. (R) is stated beside it because the relativization barrier is the older result and needs no extension. The template of Section 8.6 uses (R); Section 8.6 states what (A) and (N) add.

8.5. The Covering Lemma and the Step Premise

The barrier results exclude three families. The sector-elimination lemma needs, in addition, that the excluded families contain every proof whose engagement lies within the first two kinds. The covering lemma shows that the relativizing family alone does.

An oracle machine is a machine of the same model with one more kind of step (Definition 4.1). Its computation is again a sequence of configurations, each obtained from the one before by one application of its step rule, the oracle's answer being part of that rule. The same holds in every model I use in this book: a probabilistic oracle machine adds the oracle step to its coin flips; a quantum oracle machine applies, as one of its gates, the fixed unitary transformation the oracle defines, on a superposition of query strings; a circuit with oracle gates evaluates them, like any other gate, in order. A universal machine with access to the oracle runs every oracle machine of its model with the same overhead, answering each query by asking the oracle, as Section 4.1 states: "a machine with an oracle can still be enumerated, its steps counted, and its runs simulated by another machine with the same oracle". What an oracle machine does not keep is the inner form of a plain machine's step rule: a query step reads a whole string from the query tape and replaces it with an answer that depends on the oracle, so the bounded-window property of Turing machine steps fails for it, and an oracle gate is not among the gates whose

kinds a lower bound such as Smolensky's uses. The inner form of a machine's step rule and the kinds of a circuit's gates are exactly the facts that the boundary test puts in the third kind.

Lemma 8.6 (Covering). Let an argument establish a statement about complexity classes defined by machines or circuits of the models I use in this book. Suppose every step of the argument, and every premise it cites, is of the first or the second kind under the boundary test (Definition 6.2a). Fix any oracle, and read the statement and every step with every machine and circuit given access to that oracle. Then every step still follows and every cited premise still holds, so the argument proves the statement relative to that oracle. Since the oracle is arbitrary, the argument relativizes.

Proof. The argument is a finite derivation, and each premise it cites has a proof that is again a finite derivation, down to premises with no proof except from themselves. The proof is by induction over this finite structure: each step form of the first two kinds remains valid when every machine and circuit is given the fixed oracle, given that the premises it uses do.

(1) Checking identity and substituting equals for equals are steps of logic; they hold of any terms, whatever the oracle.

(2) Applying a definition. The definitions the argument uses are those of the models, their classes and their resources. Read with the oracle, each is the definition of the corresponding oracle model, and the classes are the relativized classes of Definition 4.2. A step that applies one of the unrelativized definitions applies the relativized one in the same way. The definition of a plain Turing machine also fixes the inner form of its step rule, the bounded window; a step that draws that form from the definition and derives from it what the machine does is of the third kind by the boundary test, and is not among the steps of this case. The bounded window of a plain Turing machine's step rule is the one place where the unrelativized and relativized definitions differ, and the boundary test has already placed a step that uses the bounded window outside the first two kinds.

(3) Running a process, directly or by a universal machine. The universal machine with access to the oracle runs every oracle machine with the same overhead, so every fact the argument draws from running a machine, including the number of steps the run takes, holds of the oracle machine.

(4) Enumerating and counting. Oracle machines are indexed as the plain machines are, with the query state added; their steps, configurations and number are counted by the same definitions, a query counting as one step. Counts of objects that do not involve machines (formulas, functions, assignments, truth tables) do not change.

(5) Building a definite object from definite inputs. A rule that builds an object from the description of a machine extends to descriptions of oracle machines by reading the query state as one more state, and every fact that follows from the rule alone holds of the extended rule. A rule that builds an object by running machines (a diagonal machine, a simulator) builds, with the oracle, the object that runs the oracle machines, by case (3).

(6) Steps of the second kind. Case analysis and conditional branching on the outputs of machines have the same cases with the oracle, since the outputs are again outputs. A query and its answer are a step of the oracle machine's computation, covered by case (3). A challenge and its response, in an interactive protocol whose verifier is a machine of a model, are steps of the verifier's computation and of the prover's, covered by cases (3) and (6).

(7) Cited premises. A cited premise of the first two kinds has a proof whose steps are of the first two kinds; by the induction its statement holds with the oracle. A premise with no proof except from itself is a definition or a primitive axiom of the models, covered by case (2).

No other step form is of the first or second kind, since Definition 6.1, with the boundary test, defines those kinds by these forms. So every step and every cited premise of the argument holds with the fixed oracle, and the argument proves its statement relative to the fixed oracle. ■

The proof uses nothing about which statement the argument proves, so the lemma holds for every argument about classes defined by machines, not only for proofs of separations.

Corollary 8.7. A statement about complexity classes that fails relative to some oracle has no proof whose steps and cited premises are all of the first two kinds. Every proof of it contains a step of the third kind, in itself or in a premise it cites.

Proof. A proof within the first two kinds relativizes by Lemma 8.6, and so proves the statement relative to every oracle, including the one relative to which it fails. ■

The corollary agrees with the record. The theorem of Cook and Levin fails relative to oracles (relative to some oracle, SAT is not complete for the relativized class NP), and its proof takes the bounded-window step. The theorem that interactive proofs decide exactly polynomial space fails relative to an oracle (Fortnow and Sipser, 1988), and its proof cites the completeness of quantified Boolean formulas, which takes the same step. The time hierarchy theorem and the halting theorem hold relative to every oracle, and the test finds their proofs within the first kind.

The covering lemma agrees with a line of results in complexity theory that reach the same conclusion from the other side. Arora, Impagliazzo and Vazirani (1992) observed that the non-relativizing results of complexity theory use one property of computation, local checkability: a computation is correct exactly when every small window of its tableau is correct, which is the property behind the theorem of Cook and Levin. They showed that the relativizing results are those provable without it, in a theory whose other axioms hold relative to every oracle. Impagliazzo, Kabanets and Kolokolova (2009) added a weaker axiom, arithmetic checkability, from which most algebrizing results follow and from which P versus NP and NP versus BPP cannot be settled. Aydinlioglu and Bach (2016) joined relativization and algebrization as affine relativization, truth relative to every oracle given by one error-correcting code applied to a language. The boundary test locates local checkability exactly: it is a fact about the inner form of a Turing machine's step rule, used to derive what every machine's accepting computations are, and so a step of the third kind.

Corollary 8.8 (Covering for a separation; Hypothesis E). Every formal, sound proof of $\mathrm{Sep}(C, D)$ that assumes no aspect of it openly, and whose engagement lies within $\{e_1, e_2\}$, relativizes; so it relativizes, algebrizes, or is natural. This statement is Hypothesis $\mathrm{E}(C, D)$.

Proof. Lemma 8.6, applied to the proof. ■

The sector-elimination lemma then gives every proof a step of the third kind. For the third kind to count toward the separation, the step must derive what the machines the separation compares do or cannot do (Definition 6.3). Section 8.2 shows that every proof contains derivations of what the machines compared do or cannot do; the step premise places the step of the third kind among those derivations.

Premise L(C, D) (The step on the machines compared). Let π be a formal, sound proof of Sep(C, D) that assumes no aspect of it openly. If e_3 is in Eng(π), then some step of the third kind, in π or in a premise π cites, derives, from what they are, that some machine of the model of C decides the language the proof separates, or that no machine of the model of D decides it.

The support for Premise L is Section 8.2: every such proof contains derivations of exactly these two forms, and a step of the third kind that reasoned only about some other process would be a detour the separation does not demand. Its counterexample is a proof of the separation whose only steps of the third kind are about processes other than the machines the separation compares.

8.6. The Template

Theorem 8.9 (The separation template). Let $D \subseteq C$ be classes as in Definition 8.1. Assume the relativization barrier (R) for the pair and Premise L(C, D). Then every formal, sound proof of Sep(C, D) that assumes no aspect of it openly contains a step of the third kind that derives, from what the machines are, that some machine of the model of C decides the language the proof separates, or that no machine of the model of D decides it; and Sep(C, D) is a third-kind proposition in the operative sense of Section 6.3.

Proof. (1) By (R), no proof in the relativizing family establishes the separation. (2) By Corollary 8.8, every proof in the class whose engagement lies within $\{e_1, e_2\}$ relativizes, and so belongs to the relativizing family. Statement (2) is the covering condition of Lemma 6.6 with one family, the relativizing family. Statement (1) says that no proof in the relativizing family establishes the separation. By Lemma 6.6, every proof in the class has e_3 in its engagement. (3) By Premise L, for every proof in the class some step of the third kind, in the proof or in a premise it cites, derives what the machines of one of the two models do or cannot do, as the separation states, so e_3 is in $\mathrm{Eng}_{\mathrm{Sep}}(\pi)$ for every proof π in the class. (4) By Definition 6.5, in the operative sense of Section 6.3, the separation is a third-kind proposition. ■

The proof is a chain of four steps. Step (1) cites the relativization barrier; step (2) applies the covering lemma and the sector-elimination lemma; step (3) applies Premise L; step (4) is the definition. Every step follows by the rules. The premises are of two kinds: proved results, namely the relativization barrier, the covering lemma and the sector-elimination lemma; and one named premise, Premise L.

The natural proofs and algebrization barriers, (N) and (A), are not used. The natural proofs and algebrization barriers say more about the third kind. By Propositions 8.3 and 8.4, natural and algebrizing techniques take steps of the third kind, and the two barriers show that a step of the third kind used only through a cheap, common test on truth tables, or only in the arithmetized form a low-degree extension preserves, cannot reach the separation. A proof of either separation needs a step of the third kind of a form none of the three barriers blocks.

8.7. How the Template Is Checked

Each ground of the template has a definite counterexample. An argument confined to the first two kinds of step under the boundary test that fails relative to some oracle would show Lemma 8.6 wrong. A proof of the separation whose only steps of the third kind are about processes other than the machines the separation compares would show Premise L wrong. A relativizing technique that takes a step of the third kind would show Proposition 8.2

wrong, and a natural or algebrizing technique that reaches beyond relativization by steps of the first two kinds alone would show Proposition 8.3 or 8.4 wrong; neither touches the deduction.

A proof of the separation that is formal, independent and sound, and that takes a step of the third kind about what the machines the separation compares can or cannot do, is not a counterexample. The template says what such a proof must contain, and says nothing against its existence.

8.8. What the Template Does Not Say

The template answers the third-kind question. The template does not apply the FIS Theorem, which requires a demonstration that every formal derivation of the separation assumes part of the separation (Section 7.3). It does not say that the separation is true, or false, or unprovable in any system. It does not say that the techniques it excludes are mistaken: every step of a relativizing, natural or algebrizing argument follows by its rules, and the barrier theorems are themselves formal, independent and sound proofs of their own claims. The template says that the relativizing, natural and algebrizing families do not reach the separation, and, under the step premise, that every proof that does reach the separation contains a step of the third kind, of a form none of those families takes.

8.9. Summary

A separation $\mathrm{Sep}(C, D)$ asserts that machines of one model decide a language that machines of the other cannot, and every proof of it derives, from what the machines are, what they can or cannot do. The covering lemma shows that every argument confined to the first two kinds of step relativizes. The template takes the relativization barrier for the pair, the covering lemma and Premise L, and by the sector-elimination lemma concludes that the separation is a third-kind proposition. The engagement bounds of the three families say what each barrier shows about the kinds of step, and the deduction does not use them. Chapters 9 and 10 apply the template to $\mathrm{BQP} \neq \mathrm{BPP}$.

PART III

The Two Classifications

Chapter 9. BQP ≠ BPP: What It Asserts and What Every Proof of It Must Show

ABSTRACT

This chapter applies the first two filters of the procedure to BQP ≠ BPP. The separation asserts that machines whose paths combine by adding complex amplitudes, which can cancel, decide in polynomial time some language that machines whose paths combine by adding nonnegative probabilities cannot. The two classes have the same bound on time and the same bound on error, and the claim is that the way the paths combine makes the difference. Every proof of it must show that a quantum circuit family decides some language and that no probabilistic machine does, and so it derives, from what each kind of machine is, what it can or cannot do. A proof is a classical object reasoning about quantum machines, and that does not affect the test, which concerns what the proof's steps are about.

9.1. What the Separation Asserts

BPP and BQP have the same bound on time, polynomial in the length of the input, and the same bound on error, one third, which repetition reduces in both cases. They differ in the machine. A probabilistic machine has a state that is a probability distribution over its configurations; its steps are stochastic matrices; a state of two systems together is a distribution over pairs of configurations, which is always a mixture of independent states of the two; and the probabilities of paths that reach the same outcome add. A quantum machine has a state that is a unit vector of complex amplitudes; its steps are unitary matrices; two systems combine by the tensor product, which has entangled states that are not mixtures of independent states of the two; and the amplitudes of paths that reach the same outcome add, and can cancel.

BQP ≠ BPP asserts that this difference changes the class of languages decided in polynomial time: that there is a language some uniform family of polynomial-size quantum circuits decides with bounded error, and that no polynomial-time probabilistic machine decides with bounded error.

The separation is not self-dependent in the sense of the FIS book (FIS Definition 10.1). The separation is not a claim about every member of a class to which every formal proof belongs, and being such a claim is the first condition of the FIS book's definition of self-dependence. So the FIS book's self-dependence theorem does not apply to BQP ≠ BPP, and in this chapter and the next I ask the third-kind question.

9.2. Criterion 1

The separation asserts that what the machines of one model are, quantum circuits whose amplitudes interfere, lets them decide a language in polynomial time that the machines of another model, probabilistic machines whose probabilities only accumulate, cannot. No resource distinguishes the two classes: the time bound, the error bound and the length of the register are the same. What distinguishes them is the kind of machine, and the claim is that the kind of machine fixes what can be done. The claim that the kind of machine fixes what can be done is a claim that what a process is limits what the process does, and so the separation passes Criterion 1. In the terms of comparison set out in Section 8.2, it asserts that context constrains content: the kind of machine falls within context, the languages it decides within content, and the form of the separation within the form of a limit of content by context.

The time hierarchy theorem gives the contrast. It separates two classes of the same deterministic machines by the amount of time they are given, and its proof builds a diagonal machine that differs from each machine of the smaller class on some input, by one rule applied alike to every machine. The difference between its classes is a quantity, and steps of the first kind establish it. The difference between BPP and BQP is the kind of machine, with every quantity held fixed.

9.3. Criterion 2

Write Sep for BQP $\neq$ BPP. Sep asserts that there is a language L such that (a) some uniform family of polynomial-size quantum circuits decides L with bounded error; and (b) no polynomial-time probabilistic machine decides L with bounded error. Every proof of Sep establishes (a) and (b) for some language, itself or by citing results that do.

A proof of (a) takes the description of a circuit family and derives the probability that the measured output bit is correct. For a language that is to separate the classes, the derivation goes through features that distinguish quantum from probabilistic computation. Shor's analysis goes through the quantum Fourier transform and the interference it produces; Simon's goes through superposed evaluations of a function and interference over a hidden subgroup. A derivation that went only through features the two models share would show only what a probabilistic machine can also do, and would not separate the classes. So the proof of (a) derives what a process does from what it is; for an algorithm such as Shor's it does so from the particular gates of the circuit, which is a step of the third kind by the test of Section 6.2.

A proof of (b) takes the class of polynomial-time probabilistic machines and concludes that none of them decides L. That is a derivation of a limit on what the members of a class can do from what they are.

A proof that cites (a) or (b) as results proved elsewhere brings their steps with it, so citation moves the derivations and does not remove them. Every proof of Sep therefore derives, from what the machines are, what quantum machines can do or what probabilistic machines cannot, in the specific form Sep asserts it. Sep passes Criterion 2.

What Sep passes under Criterion 2 is not the general fact that every proof is a process whose form fixes its output. That holds of a proof of the Pythagorean theorem as much as of a proof of Sep, and it tells nothing apart. For Sep, the relation between the kind of machine and what it can do is the object of the proof's steps.

9.4. A Classical Proof About Quantum Machines

A formal proof is a classical object: a finite sequence of symbols, checked step by step by a classical procedure. It does not superpose, entangle or interfere. A proof of Sep is a classical object that establishes something about quantum machines, and the question arises whether that affects the test.

That a proof of Sep is a classical object does not affect the test. Criterion 2 asks what the proof's steps are about, not what the proof is made of. A proof that water expands when it freezes need not freeze; its steps must still reason about the relation between the structure of ice and its volume. A classical proof that a quantum circuit family decides a language computes amplitudes, bounds interference terms and derives the output distribution: it reasons about superposition and interference without doing either. In the same way a proof that a given Turing machine halts on a given input is not a Turing machine; its steps take the machine's table as given and derive what the machine does.

The point also shows what the test does not say. That every proof of Sep derives what these machines can or cannot do from what they are is a statement of what every proof must do. The requirement that every proof derive what these machines can or cannot do is not a statement that any proof assumes what that proof proves. The proof of (a) can take as premises the description of the circuits and the rules of quantum mechanics as a model of computation; the proof of (b) can take as premises the definition of probabilistic machines; and no one of these premises, by itself, implies Sep or any part of it.

9.5. What the Filters Establish

Criteria 1 and 2 together give a definite profile. The separation asserts a relation between what machines are and what they can do, and every proof of it derives a capacity of quantum machines from what they are and a limit of probabilistic machines from what they are.

Criteria 1 and 2 are necessary conditions for a proposition to require a step of the third kind, but they are not sufficient conditions. A proposition can pass both and still be Type S or Type P, if every step a proof takes about what

the machines can and cannot do can be done by steps of the first two kinds. Whether that is possible for Sep is the question of Criterion 3, which the next chapter answers.

9.6. Summary

BQP $\neq$ BPP asserts that the way a machine's paths combine, by amplitudes that can cancel or by probabilities that only add, changes what it decides in polynomial time, with every resource held fixed. It passes Criterion 1. Every proof of it derives what some quantum circuits do from what they are and a limit on probabilistic machines from what they are, and so it passes Criterion 2. Both are statements of what every proof must do, and neither is a statement that any proof assumes part of the separation.

Chapter 10. BQP ≠ BPP: The Third Kind

ABSTRACT

This chapter applies Criterion 3, the sector-elimination argument, to BQP ≠ BPP, through the template of Chapter 8. The barrier results for the pair are proved for BQP and BPP themselves: a PSPACE-complete oracle makes them equal, the same oracle with its multilinear extension makes them equal in the algebrized setting, and the natural proofs barrier blocks the route through a circuit lower bound. With the covering lemma of Chapter 8 and the step premise Premise L(BQP, BPP), the template gives Theorem 10.1: every proof of the separation contains a step of the third kind that derives what quantum circuits do, or a limit on what probabilistic machines can do, from what they are. I set out what the covering lemma says of the known techniques, the place of Williams's method, the place of the pseudorandomness assumption, and what the classification does not claim.

10.1. The Barrier Results for BQP and BPP

The template uses the relativization barrier for the pair; the other two barrier results show which steps of the third kind are not enough (Section 8.4).

(R) Let A be complete for PSPACE. A quantum polynomial-time computation with access to A can be simulated in polynomial space, since each amplitude is a sum over paths computed one path at a time and each query is answered in polynomial space (Bernstein and Vazirani, 1997); so $\mathrm{BQP}^A \subseteq \mathrm{PSPACE} \subseteq \mathrm{P}^A \subseteq \mathrm{BPP}^A$. The containment of BQP in BPP relative to the PSPACE-complete oracle is (R) for the pair (Theorem 4.4).

(A) Let A again be complete for PSPACE, and $\tilde{A}$ its multilinear extension. The multilinear extension of a language in PSPACE is itself computable in polynomial space (Babai, Fortnow and Lund, 1991), so a quantum polynomial-time computation with access to $\tilde{A}$ is simulated in polynomial space, and

$\mathrm{BQP}^{\tilde{A}} \subseteq \mathrm{PSPACE} \subseteq \mathrm{BPP}^{A}$. The containment of BQP, with access to the multilinear extension, in BPP, with access to the PSPACE-complete oracle, is (A) for the pair (Theorem 4.10(i)).

(N) BPP is contained in P/poly (Adleman, 1978; Bennett and Gill, 1981), so a language in BQP that is not in P/poly is not in BPP. Under the existence of pseudorandom function generators secure against circuits of size $2^{n^{\varepsilon}}$, no natural proof shows that any language needs superpolynomial circuits (Theorem 4.7), and so no natural proof completes this route to the separation.

The other halves of the barriers, which block proofs of BQP = BPP, are also proved for the pair: the oracles of Bernstein and Vazirani (1997) and of Simon (1997), and the pair of Aaronson and Wigderson's Theorem 5.11, built from the exponential separation of quantum from randomized communication complexity due to Raz (1999). These results use features of quantum computation, superposed queries and quantum communication, that have no counterpart in the barrier results for P and NP. The barrier results for BQP and BPP are therefore theorems about these two classes, not consequences of the results for P and NP, and the classification below rests on them.

10.2. The Covering Statement and the Step Premise

Corollary 8.8 gives, for this pair, Hypothesis E(BQP, BPP): every formal, sound proof of BQP ≠ BPP that assumes no aspect of it openly, and whose engagement lies within $\{e_1, e_2\}$, relativizes.

> **Premise L(BQP, BPP).** Let π be a formal, sound proof of BQP ≠ BPP that assumes no aspect of it openly. If e_3 is in $\mathrm{Eng}(\pi)$, then some step of the third kind, in π or in a premise π cites, derives, from what they are, that some polynomial-size quantum circuits decide the language π separates, or that no probabilistic polynomial-time machine decides it.

Premise L is supported by Chapter 9, which shows that every such proof contains derivations of exactly these two forms. The covering statement is proved (Lemma 8.6), and Section 10.4 sets out what it says of the known techniques.

10.3. The Classification

Theorem 10.1 (Classification of BQP ≠ BPP). Assume Premise L(BQP, BPP). Then every formal, sound proof of BQP ≠ BPP that assumes no aspect of it openly contains a step of the third kind that derives what quantum circuits do from what they are, or a limit on what probabilistic machines can do from what they are. So BQP ≠ BPP is a third-kind proposition.

Proof. BPP ⊆ BQP, so the pair satisfies the hypotheses of Theorem 8.9. Section 10.1 gives (R). Theorem 8.9, with Premise L(BQP, BPP), gives the conclusion. ■

The classification rests on the relativization barrier, which is unconditional, on the covering lemma, which is proved, and on Premise L. The algebrization barrier, which is also unconditional, and the natural proofs barrier, which rests on the pseudorandomness assumption, show which steps of the third kind are not enough (Section 8.6).

10.4. What the Covering Lemma Says of the Known Techniques

Three observations show what the covering lemma says of the known techniques.

Separate discovery. The three barriers were found over thirty-four years by different people studying different methods: oracle constructions in 1975, circuit lower bounds in 1997, arithmetization in 2009. None was built to complement the others. The covering lemma shows why the first of the three barriers, relativization, reaches every technique confined to the first two kinds, and Propositions 8.3 and 8.4 show that the other two barriers block two forms of the third kind.

The known techniques. The known methods for separating complexity classes fall into the three families, and Williams's algorithmic method is the principal exception. Diagonalization and simulation relativize, and so do the hierarchy theorems proved by them. The lower bounds against restricted circuits, by random restrictions, switching lemmas, the approximation method and the polynomial method of Razborov and Smolensky, are natural. Arithmetization and the results built on it, IP = PSPACE and MIP = NEXP among

them, algebrize (Aaronson and Wigderson, 2009, Theorems 3.7 and 3.8). A technique assembled from parts each confined to the first two kinds is itself confined to them, since the engagement of a proof is the set of kinds its steps use, and by the covering lemma it relativizes.

Williams's method. The algorithmic method of Williams (2014) lies outside all three families and has given new lower bounds: it proved that nondeterministic exponential time is not contained in ACC^0, the class of constant-depth circuits with counting gates modulo any fixed number. The algorithmic method of Williams does not relativize, is not natural, and does not algebrize. The proofs of Williams's method are not proofs of BQP $\neq$ BPP. By the engagement template (Definition 6.2), the method takes a step of the third kind. The method's decisive step designs a satisfiability algorithm for the circuit class, faster than trying every input, by taking what the circuits of the class are, for ACC^0 their representation as a symmetric function of many small conjunctions (Beigel and Tarui, 1994), and deriving from it a faster way to decide what they compute. That step takes as its object what the gates of the circuits are and what they make the circuits compute, which the boundary test assigns to the third kind. The lower bound then follows by combining the algorithm with the nondeterministic time hierarchy theorem. So the method takes a step of the third kind, about what the gates of a circuit class make its circuits compute, of a form that none of the three barriers blocks.

A counterexample to the covering lemma would be an argument confined to the first two kinds of step that fails relative to some oracle, and the covering lemma shows that no such argument exists.

10.5. The Pseudorandomness Assumption

The natural proofs barrier assumes pseudorandom function generators secure against circuits of size 2^{n^ε}. They follow from one-way functions of the corresponding strength, and their existence is a standard assumption of cryptography: candidate one-way functions from factoring, discrete logarithms and lattice problems have resisted every known attack, and for the lattice problems the best known algorithms take exponential time.

The assumption does not enter the classification. It enters the natural proofs barrier, which shows that a step of the third kind used only through a cheap, common test on truth tables cannot give a superpolynomial circuit lower bound, and so cannot give the separation by that route.

10.6. What the Classification Means

The classification explains the barriers. Relativization eliminates every technique confined to the first two kinds of step, by the covering lemma, and the other two eliminate two forms of the third kind. What any proof of the separation needs, and what the eliminated families lack, is a step that reasons from what quantum circuits are to what they can do, or from what probabilistic machines are to what they cannot do, whose work no step of the first two kinds can do.

The classification also says where to look for a proof. The question for a proof is not which refinement of a relativizing, natural or algebrizing technique might work; given the barrier theorems, none will. The question for a proof is what step of the third kind could carry the separation: what derivation, from what each kind of machine is to what that kind can do, could tell the two kinds apart. Williams's method is the known example of a technique that takes such a step.

The classification does not say that the separation is true or false, or that it is unprovable in any system. The classification does not say that any proof of the separation must assume part of the separation. A proof that derives the separation from the definitions of the two models, taking a step of the third kind about what quantum circuits and probabilistic machines can and cannot do, would be consistent with every result of this book, and it could be formal, independent and sound.

10.7. Summary

The barrier results for BQP and BPP are proved for those classes. With the covering lemma and Premise L(BQP, BPP), the template gives Theorem 10.1: every proof of BQP $\neq$ BPP contains a step of the third kind that derives what quantum circuits do, or a limit on what probabilistic machines can do, from

what they are, and the separation is a third-kind proposition. The classification does not use the pseudorandomness assumption. Williams's method, a known technique outside the three families, takes a step of the third kind. Chapters 11 and 12 apply the template to $P \neq NP$.

Chapter 11. P ≠ NP: What It Asserts and What Every Proof of It Must Show

ABSTRACT

This chapter applies the first two filters of the procedure to P ≠ NP. The separation asserts that deterministic machines running in polynomial time cannot decide some language whose yes instances have witnesses that such machines check. The time bound is the same on both sides, and the claim is that the way the machine branches, along one path or by the existence of an accepting path, makes the difference. Every proof of it must show, for some language in NP, that no deterministic polynomial-time machine decides it, and so it derives, from what those machines are, a limit on what they can do. I separate that relation from the general fact that every deterministic process has its output fixed by its form, which holds whether or not P equals NP.

In this chapter and the next I apply to P ≠ NP the steps that I apply to BQP ≠ BPP in Chapters 9 and 10, in the same order, so that the two classifications can be read side by side; where the wording is the same, the argument is the same.

11.1. What the Separation Asserts

P ≠ NP asserts that there is a language L in NP that no deterministic polynomial-time machine decides. By the theorem of Cook and Levin it is equivalent to the statement that SAT is not in P. Spelled out, it says that for every deterministic machine M and every polynomial p, either M takes more than $p(n)$ steps on some input of some length n, or M answers wrongly on some input.

The statement concerns every input length at once. For each fixed length there is a machine that is correct on every formula of that length and fast: one that reads the answer from a table. The table for length n has exponentially

many entries, and no machine of fixed size carries the tables for all lengths. So P ≠ NP is a statement about growth as the input length increases without bound, and no check of instances of bounded size settles it.

The two classes have the same machines in one sense and different machines in another. NP is defined by a deterministic polynomial-time verifier together with an existential quantifier over witnesses of polynomial length, or, equivalently, by nondeterministic polynomial-time machines, which accept when some path accepts. P is defined by deterministic polynomial-time machines, which must reach their answer along one path. The time bound is the same; the difference is the mode of branching.

The separation is not self-dependent in the sense of the FIS book. The separation is not a claim about every member of a class to which every formal proof belongs, and being such a claim is the first condition of the FIS book's definition of self-dependence. So the self-dependence theorem does not apply to P ≠ NP, and in this chapter and the next I ask the third-kind question.

11.2. Criterion 1

P ≠ NP asserts that what deterministic polynomial-time machines are, machines that follow one path within polynomially many steps, limits what they can decide, so that some language decided with the help of an existential quantifier over witnesses is out of their reach. The assertion of P ≠ NP is a claim that what a class of processes is limits what the members of that class do, and so the separation passes Criterion 1.

In the terms of comparison set out in Section 8.2, P ≠ NP asserts that context constrains content: that what a process is, its structure, limits what it does, its function. The match of P ≠ NP to a limit of content by context is a structural equivalence, and both inclusions of that structural equivalence are shown. The time bound and the deterministic mode of branching that define the class fall within the definition of context, what the machines are; deciding a language falls within the definition of content, what the machines do; and the form of P ≠ NP, that some language is beyond every machine of the class, falls within the form of a limit of content by context. A structural equivalence with its inclusions shown establishes its result as a theorem does, so P ≠ NP is, exactly, a claim that context constrains content.

The claim must be read exactly. That a deterministic machine's table fixes its output on every input is true of every deterministic machine, and it would be just as true if P equaled NP: a polynomial-time machine that decided SAT would also be a machine whose table fixed its output. So that general fact is not what P $\neq$ NP asserts, and it is not an aspect of P $\neq$ NP. The two claims have different forms. That a machine's table fixes its output is a claim of determination: what a process is fixes what it does, on every input. P $\neq$ NP is a claim of limitation: some language lies beyond every process of the class. The form of the claim of determination does not fall within the form of the claim of limitation, and so a proof's being a definite process, whose structure fixes its output, is not an instance of what P $\neq$ NP asserts. The time hierarchy theorem shows the same: it asserts that a time bound limits what can be decided, which is context constraining content, and its proof, a definite process like every proof, is formal, independent and sound. What P $\neq$ NP asserts is the specific limit: that no machine of the class, whatever its table, decides SAT within a polynomial number of steps.

The contrast with the time hierarchy theorem is the same as for BQP and BPP. The hierarchy theorem separates two classes of the same deterministic machines by the time they are given, and steps of the first kind establish it. P and NP have the same time bound and differ in the mode of branching.

11.3. Criterion 2

Write Sep for P $\neq$ NP. Sep asserts that there is a language L such that (a) some nondeterministic polynomial-time machine decides L, or equivalently some deterministic polynomial-time verifier checks witnesses for it; and (b) no deterministic polynomial-time machine decides L. Every proof of Sep establishes (a) and (b) for some language, itself or by citing results that do.

For a proof that uses SAT, part (a) is short: the verifier that evaluates a formula on a proposed assignment checks the assignment in polynomial time, and the proof that it does applies the definition of an assignment satisfying a formula and counts the verifier's steps: a derivation of what the verifier does, in the sense of Criterion 2, made by steps of the first kind under the test of

Section 6.2, unlike the proof that a given program sorts, whose central step is of the third kind. A proof that uses another language in NP needs a corresponding verifier and its analysis.

Part (b) carries the weight. A proof of (b) takes the class of deterministic polynomial-time machines and concludes that none of them decides L: a derivation of a limit on what the members of a class can do from what they are. A proof that cites a hardness result for L brings that result's steps with it. And a proof that establishes Sep through a stronger statement, such as a circuit lower bound for a language in NP, derives a limit on what small circuits can do from what they are, and with $\mathrm{P} \subseteq \mathrm{P/poly}$ concludes (b).

Every proof of Sep therefore derives, from what the machines are, what the verifier or nondeterministic machine does or what the deterministic polynomial-time machines cannot do, in the form Sep asserts it. Sep passes Criterion 2.

As in the quantum case, this is not the general fact that every proof is a process whose form fixes its output. It is the specific fact that Sep is a conjunction of a capacity of one model and a limit on another, and that a proof of a conjunction establishes each part.

11.4. Working on a Relation Is Not Assuming It

Every proof of Sep must contain a derivation of a limit on what deterministic polynomial-time machines can do. The requirement that every proof of Sep contain a derivation of a limit is a statement of what every proof must do. It is not a statement that any proof assumes part of Sep. A proof of (b) can take as premises the definition of a deterministic Turing machine, the definition of polynomial time and the definition of the language, and no one of these premises, taken alone, implies Sep or any part of it: the definition of a deterministic machine, for instance, says how such a machine runs and nothing about which languages it decides in polynomial time.

A proved result of the same shape shows this. Paul, Pippenger, Szemerédi and Trotter (1983) proved that nondeterministic linear time is strictly more powerful than deterministic linear time, $\mathrm{DTIME}(n) \neq \mathrm{NTIME}(n)$, for multitape Turing machines. Their proof derives, from what deterministic linear-time machines are, a limit on what they can do: the computation of such a

machine can be cut into blocks whose dependencies form a graph of a special kind (its step of the third kind: from the particular step rule of multitape machines, each step changing only the cells under its heads, it derives that every deterministic linear-time computation can be so cut), and from this a deterministic linear-time computation is simulated by an alternating machine with four blocks of quantifiers in time slightly less than linear; if deterministic and nondeterministic linear time were equal, the four blocks would collapse to none, and the time hierarchy for alternating machines would be contradicted. The proof by Paul, Pippenger, Szemerédi and Trotter is formal, the premises of the proof are the definitions of deterministic and nondeterministic multitape Turing machines, and the proof is sound. It derives, from the mode of branching of each kind of machine, what can be done in linear time, as every proof of Sep must derive what can be done in polynomial time, and it assumes nothing it proves.

11.5. Summary

P ≠ NP asserts that deterministic machines limited to polynomially many steps cannot decide some language whose witnesses they can check, with the time bound the same on both sides. P ≠ NP passes Criterion 1, and the relation that P ≠ NP asserts is the specific limit, not the general fact that a machine's table fixes its output. Every proof of it derives a limit on what deterministic polynomial-time machines can do from what they are, and so it passes Criterion 2. The proved separation of deterministic from nondeterministic linear time shows that such a derivation can be formal, independent and sound. Chapter 12 applies Criterion 3.

Chapter 12. P ≠ NP: The Third Kind

ABSTRACT

This chapter applies Criterion 3 to P ≠ NP through the template of Chapter 8. The barrier results for the pair are those of Baker, Gill and Solovay, Aaronson and Wigderson, and Razborov and Rudich. With the covering lemma of Chapter 8 and the step premise Premise L(NP, P), the template gives Theorem 12.1: every proof of P ≠ NP contains a step of the third kind that derives a limit on what deterministic polynomial-time machines can do, or what nondeterministic machines or verifiers do, from what they are. I set out what the covering lemma says of the known techniques, the place of Williams's method and of the lower bounds that followed it, and why the pseudorandomness assumption, which itself implies P ≠ NP, is not used by the classification and would not make the classification circular.

I apply to P ≠ NP the steps that Chapter 10 applies to BQP ≠ BPP, in the same order; where the wording is the same, the argument is the same.

12.1. The Barrier Results for P and NP

(R) Let A be complete for PSPACE. Then $\mathrm{NP}^A \subseteq \mathrm{PSPACE} \subseteq \mathrm{P}^A$, so $\mathrm{NP}^A = \mathrm{P}^A$ (Baker, Gill and Solovay, 1975; Theorem 4.3).

(A) Let A be complete for PSPACE and $\tilde{A}$ its multilinear extension. Then $\mathrm{NP}^{\tilde{A}} \subseteq \mathrm{PSPACE} \subseteq \mathrm{P}^A$ (Aaronson and Wigderson, 2009; Theorem 4.10(i)).

(N) P is contained in P/poly, so a language in NP that needs superpolynomial circuits is not in P. Under the existence of pseudorandom function generators secure against circuits of size 2^{n^ε}, no natural proof shows that any language needs superpolynomial circuits (Theorem 4.7).

The other halves, which block proofs of P = NP, are the oracle of Baker, Gill and Solovay relative to which P and NP differ, and the pair of Aaronson and Wigderson relative to which $\mathrm{NP}^B \not\subseteq \mathrm{BPP}^{\tilde{B}}$. These results are proved for P and NP. They are independent of the results of Chapter 10: the separating or-

acle for P and NP is built by a counting argument with no quantum content, and the quantum oracles of Chapter 10 have no use here. The template uses (R); (A) and (N) show which steps of the third kind are not enough.

12.2. The Covering Statement and the Step Premise

Corollary 8.8 gives, for this pair, Hypothesis E(NP, P): every formal, sound proof of P ≠ NP that assumes no aspect of it openly, and whose engagement lies within $\{e_1, e_2\}$, relativizes.

> **Premise L(NP, P).** Let π be a formal, sound proof of P ≠ NP that assumes no aspect of it openly. If e_3 is in $\mathrm{Eng}(\pi)$, then some step of the third kind, in π or in a premise π cites, derives, from what they are, what nondeterministic polynomial-time machines or polynomial-time verifiers do on the language π separates, or that no deterministic polynomial-time machine, or no polynomial-size circuit that simulates one, decides it.

Premise L is supported by Chapter 11, which shows that every such proof contains a derivation of a limit on deterministic polynomial-time machines, directly or through circuits. The circuits are included because a proof by a circuit lower bound derives a limit on small circuits and concludes the limit on machines through P ⊆ P/poly.

12.3. The Classification

> **Theorem 12.1 (Classification of P ≠ NP).** Assume Premise L(NP, P). Then every formal, sound proof of P ≠ NP that assumes no aspect of it openly contains a step of the third kind that derives what nondeterministic polynomial-time machines or verifiers do from what they are, or a limit on what deterministic polynomial-time machines, or the polynomial-size circuits that simulate them, can do from what they are. So P ≠ NP is a third-kind proposition.

Proof. P ⊆ NP, so the pair satisfies the hypotheses of Theorem 8.9. Section 12.1 gives (R). Theorem 8.9, with Premise L(NP, P), gives the conclusion. ■

Theorems 10.1 and 12.1 are two instances of one template. Each rests on the relativization barrier proved for its own pair of classes, on the covering lemma, and on a step premise stated for its own separation. Neither classification is a corollary of the other: the template is shared, and each conclusion follows from the facts of its own case.

12.4. What the Covering Lemma Says of the Known Techniques

What Section 10.4 says of the known techniques holds for P and NP. The known methods for separating complexity classes fall into the three families, and the principal exception, Williams's algorithmic method, takes a step of the third kind (Section 10.4).

For P and NP, Williams (2014) proved that NEXP is not contained in ACC^0. Murray and Williams (2018) extended the bound to nondeterministic quasi-polynomial time. Each derives a lower bound from an algorithm built on the gates of the circuits of a class, and each proves its bound for a nondeterministic class above NP, against the restricted circuits of ACC^0. The strongest circuit lower bounds for an explicit class come from a different method. Chen, Hirahara and Ren (2024) proved that symmetric exponential time with one bit of advice, a class above the polynomial hierarchy, contains a language that needs circuits of size close to the maximum, $2^n/n$, and Li (2024) removed the bit of advice, by an algorithm for the range avoidance problem: given a circuit that maps n bits to more than n bits, find a string outside its range. Chen, Hirahara and Ren state that their proofs relativize, so their method lies in the relativizing family, and it reaches a class far above NP and, by the relativization barrier, cannot reach $\mathrm{P} \neq \mathrm{NP}$. A proof of $\mathrm{P} \neq \mathrm{NP}$ by Williams's method would be a proof of the kind the classification describes: one that contains a step of the third kind deriving, from what small circuits are, a limit on what they can compute.

By the covering lemma there is no proof of $\mathrm{P} \neq \mathrm{NP}$, and no technique able to give one, whose engagement lies within the first two kinds.

12.5. The Pseudorandomness Assumption

For P ≠ NP the pseudorandomness assumption has a feature it lacks for BQP ≠ BPP. Strong pseudorandom function generators give one-way functions, and one-way functions exist only if P ≠ NP. So the assumption on which the natural proofs barrier rests implies the separation being classified.

The classification does not use the assumption: Theorem 12.1 rests on the relativization barrier, the covering lemma and Premise L. Were the assumption a premise of the classification, it would still not make it circular. The classification is not a proof of P ≠ NP. It is a statement about every proof of P ≠ NP: what such a proof contains. The assumption is a premise of that statement, and the statement does not conclude P ≠ NP; it concludes that every proof of P ≠ NP in the class contains a step of the third kind. A premise of a classification that implies the proposition classified is not a premise of any proof of that proposition.

The assumption enters the book at the natural proofs barrier, which shows that a step of the third kind used only through a cheap, common test on truth tables cannot give a superpolynomial circuit lower bound; that barrier explains, and the classification does not rest on it.

12.6. What the Classification Means

The classification explains the shape that the work on P ≠ NP has taken. The techniques that settled other separations, diagonalization and simulation, are relativizing. The techniques that gave strong lower bounds against restricted circuits are natural. Arithmetization, the method that crossed the relativization barrier, algebrizes. None of these families can reach P ≠ NP (the natural family under the pseudorandomness assumption), and, under the step premise, every proof that reaches it contains a step of the third kind that derives a limit on what deterministic polynomial-time machines can do, or what nondeterministic machines or verifiers do, from what they are, of a form none of these families takes.

The classification does not say that P ≠ NP is true, or false, or unprovable in any system. It does not say that any proof of P ≠ NP must assume part of it. A proof that derives the separation from the definitions of the two models,

taking a step of the third kind about what deterministic and nondeterministic machines can and cannot do, would be consistent with every result of this book, and it could be formal, independent and sound. The separation of deterministic from nondeterministic linear time (Section 11.4) is a proved result of that kind at a smaller scale.

12.7. Summary

The barrier results for P and NP are proved for those classes. With the covering lemma and Premise L(NP, P), the template gives Theorem 12.1: every proof of P $\neq$ NP contains a step of the third kind that derives a limit on what deterministic polynomial-time machines can do, or what nondeterministic machines or verifiers do, from what they are, and P $\neq$ NP is a third-kind proposition. The pseudorandomness assumption implies P $\neq$ NP; the classification does not use it, and would not be circular if it did, since the classification is a statement about the proofs and not a proof of the separation. Chapter 13 records the results that bear on both separations.

Chapter 13. The Record

ABSTRACT

The fourth criterion of the procedure records what the literature on a proposition shows and classifies nothing. For each separation the record has three entries. The barrier theorems are formal, independent and sound proofs that whole families of techniques do not settle it. A body of formal, sound results bears on each separation without proving that separation, among them a proved separation of the same shape at a smaller scale: for BQP and BPP, the unconditional separation of shallow quantum and classical circuits of Bravyi, Gosset and König; for P and NP, the separation of deterministic from nondeterministic linear time of Paul, Pippenger, Szemerédi and Trotter. And separate lines of evidence converge on each separation without a formal derivation of it. The record holds no formal, sound proof that assumes part of either separation. It is consistent with the classifications of Chapters 10 and 12, and it does not show that either separation is inescapably circular.

13.1. The Criterion

Criterion 4 (The configuration record; FIS Section 12.7). Record, for the results that bear on *Prop*, which of three configurations they occupy: (a) formal arguments that assume nothing about *Prop* and do not establish *Prop*; (b) formal, sound proofs of *Prop* that assume part of *Prop*; (c) independent and sound knowledge of *Prop* held without a formal derivation.

For a proposition shown to be inescapably circular, the FIS Theorem predicts that every approach falls under (a), (b) or (c), and the record shows under which of (a), (b) and (c) each approach falls. For a proposition not shown to be inescapably circular, such as the two separations of this book, the record describes the literature and supports nothing about circularity, since barriers, disputes and convergent evidence can arise from difficulty alone (FIS Section 11.6).

13.2. The Barriers

The barrier theorems are formal, independent and sound proofs of their own claims. Each shows that a family of techniques does not settle the separations; the techniques in each family are formal arguments whose steps follow by their rules and whose premises assume nothing about the separations. Each family is therefore an instance of configuration (a) for both separations. The failure of each family of techniques is evidence about the kinds of step a proof needs, which is the evidence that the classifications use. It is not evidence that any proof must assume part of a separation.

13.3. Results Bearing on BQP ≠ BPP

Shor's algorithm (1994) shows that factoring is in BQP. Its proof derives, from the quantum Fourier transform and the interference it produces, that the algorithm finds the period of a modular exponential function in polynomial time. The premises of the proof of Shor's algorithm are the definition of the circuit model and elementary number theory, and the proof assumes nothing that the proof itself proves. It does not prove the separation, which would also need factoring to be outside BPP.

Simon's theorem (1997) and the oracle separation of **Raz and Tal** (2019), BQP $\not\subseteq$ PH relative to an oracle drawn from a distribution built on the forrelation problem, derive from what quantum query algorithms are what they can detect by interference, and from what classical query algorithms are what they cannot. Constructing an oracle is not assuming a conclusion: the premises are the definitions of the classes and of the oracle, and the conclusions are relativized separations.

Quantum walks. The walk on the glued-trees graph of Childs, Cleve, Deotto, Farhi, Gutmann and Spielman (2003) crosses between two marked vertices in polynomial time, where every classical algorithm that explores the graph by queries needs exponential time. Both halves are derived from the definitions of the two kinds of algorithm.

Sampling. The results of Aaronson and Arkhipov (2011) and Bremner, Jozsa and Shepherd (2011) derive the output distribution of a quantum device from what it is, and connect its classical simulation to a collapse of the polynomial hierarchy. The hardness of the classical simulation rests on stated hypotheses, which are premises distinct from the results.

Shallow circuits. Bravyi, Gosset and König (2018) proved, without an oracle and without any assumption, that constant-depth quantum circuits solve a problem that constant-depth classical circuits with gates of bounded fan-in cannot. The problem is the two-dimensional hidden linear function problem: given a binary quadratic form on a grid of bits, return one of the strings the problem defines as correct. A quantum circuit of constant depth, with gates acting on neighboring quantum bits of a grid, returns a correct string with certainty. Any classical probabilistic circuit whose gates have bounded fan-in and which returns a correct string with high probability must have depth growing with the logarithm of the input size. The upper bound derives a capacity from a structure: from the circuit's gates and the entangled state they prepare, every outcome of the measurement is correct. The lower bound derives a limit from a structure: in a classical circuit of depth d with gates of fan-in at most K, each output bit depends on at most K^d input bits, and outputs so constrained cannot reproduce the correlations the quantum outputs satisfy, which are of the kind studied in nonlocal games. The proof's premises are the definitions of the two circuit models and of the problem; none of them by itself implies the separation or any part of it. It is a formal, independent and sound proof of a separation of quantum from classical circuits, of the same shape as BQP $\neq$ BPP, for a different pair of classes and for a relation problem rather than a language.

13.4. Results Bearing on P $\neq$ NP

NP-completeness. The theorem of Cook and Levin derives, from what a nondeterministic polynomial-time machine is, a polynomial-time translation of its computations into Boolean formulas, so that SAT is NP-complete. It reduces P $\neq$ NP to the single statement that SAT is not in P, and it assumes nothing about whether that statement is true. Its claim that the translation is

correct is a step of the third kind by the test of Section 6.2, and the theorem fails relative to some oracles, as Corollary 8.7 requires of a theorem whose every proof takes such a step.

Linear time. Paul, Pippenger, Szemerédi and Trotter (1983) proved that $\mathrm{DTIME}(n) \neq \mathrm{NTIME}(n)$ for multitape Turing machines (Section 11.4). It is a formal, independent and sound proof of a separation of deterministic from nondeterministic computation, of the same shape as $\mathrm{P} \neq \mathrm{NP}$, for linear time.

Restricted circuits. Razborov (1985) proved superpolynomial lower bounds for monotone circuits computing the clique function; Furst, Saxe and Sipser (1984) and Ajtai (1983) proved that parity needs superpolynomial size in constant depth, and Håstad (1987) made the bound exponential; Razborov (1987) and Smolensky (1987) proved lower bounds for constant-depth circuits with counting gates modulo a prime. Each derives a limit on what the circuits of a restricted class can compute from what those circuits are, a step of the third kind by the test of Section 6.2 (Proposition 8.3). The lower bounds against restricted circuits are the results that the natural proofs barrier shows cannot be extended to general circuits by the same methods.

Williams and after. $\mathrm{NEXP} \not\subseteq \mathrm{ACC}^0$ (Williams, 2014) and its extension to nondeterministic quasi-polynomial time (Murray and Williams, 2018) are formal, sound proofs of their own propositions, and each derives a lower bound from an algorithm built on the gates of the circuits of a class. The near-maximum circuit lower bounds for symmetric exponential time, with one bit of advice (Chen, Hirahara and Ren, 2024) and without it (Li, 2024), are formal, sound proofs by a relativizing method, for a class above the polynomial hierarchy.

13.5. Convergent Evidence

For each separation, the evidence of Chapter 3 comes from separate lines, each sound within its own domain, none assuming the separation, and none a formal derivation of it. For $\mathrm{P} \neq \mathrm{NP}$ it is the failure of independent searches across thousands of NP-complete problems and the consequences $\mathrm{P} = \mathrm{NP}$ would have. For $\mathrm{BQP} \neq \mathrm{BPP}$ it is the algorithms with large speedups, the oracle and conditional separations, the hardness of sampling and the experi-

ments. This evidence is not a consequence of any limit shown by the FIS Theorem, since none is shown for either separation, and a proof of either separation with all three properties would be consistent with it.

13.6. What the Record Shows

The record holds no formal, sound proof that assumes part of either separation, so configuration (b) has no entries for either. It contains many formal, sound and independent proofs of neighboring propositions, and these reason about the same things the separations do: what quantum circuits can do and what classical circuits cannot, what nondeterministic machines can do and what deterministic ones cannot, at smaller scales or relative to oracles. Many formal, sound and independent proofs of neighboring propositions are what the classifications lead one to expect: proofs that derive, from what machines are, what machines can and cannot do, and that assume nothing about the answer.

The two proved separations of the same shape, of shallow quantum from shallow classical circuits and of nondeterministic from deterministic linear time, show proofs of that kind most clearly. Each derives what the machines of one model can do, and what those of the other cannot, from what the two models are, as Chapters 9 and 11 find every proof of the larger separation must, and each is independent. Deriving what machines can do from what they are is not assuming what is to be proved. This is why the third-kind classifications, which concern what proofs must do, do not apply the FIS Theorem, which concerns what proofs must assume.

13.7. Summary

For both separations the record has three entries: the barrier theorems, formal and sound results on neighboring propositions, and convergent evidence. It holds no formal, sound proof that assumes part of either separation. Among the results on neighboring propositions are two proved separations of the same shape at smaller scales, each formal, independent and sound. The record is consistent with the classifications and supports no conclusion about circularity. Chapter 14 answers the questions readers ask.

Chapter 14. Questions and Answers

ABSTRACT

This chapter answers ten questions readers ask about the classifications of Chapters 10 and 12, each stated in its strongest form. The ten questions ask whether one classification is a corollary of the other; whether conditional separations, oracle separations, or the proved separations at smaller scales are counterexamples; whether a new technique could evade all three barriers; how the quantum classification relates to the extended Church–Turing thesis; what standing the engagement bounds have; what geometric complexity theory would do; whether the pseudorandomness assumption makes the classification of P ≠ NP circular; and whether the classifications say that the separations cannot be proved.

14.1. Is One Classification a Corollary of the Other?

Question. The two separations face the same three barriers and are classified by the same template. The classification of one is a corollary of the other, or both are corollaries of something more general that the book has not shown.

Answer. The template is shared and the inputs are not. P ≠ NP and BQP ≠ BPP are not known to imply each other in either direction. The barrier results for each pair are proved for that pair: the separating oracle for P and NP is built by a counting argument with no quantum content, and the separating oracles and algebrized worlds for BQP and BPP are built from quantum query and communication results that have no use for P and NP. The covering and step premises are stated for each separation separately, and each could fail while the other holds. Each classification follows from the facts of its own case through the same method, as two animals are classified as mammals by the same criteria without one classification following from the other.

14.2. What About Conditional Separations?

Question. If factoring is not in BPP, then BQP ≠ BPP, by Shor's algorithm. If some language in NP needs superpolynomial circuits, then P ≠ NP. The two conditional proofs are formal and sound, and they reach the separations by ordinary means.

Answer. Each is a proof of a conditional proposition, "if H, then the separation", which is a different proposition from the separation. Read as an argument for the separation itself, each contains what the classification says every proof contains. The quantum argument cites Shor's theorem, whose proof derives what certain quantum circuits do from what they are, and its hypothesis states a limit on what every probabilistic machine can do. The classical argument rests on a hypothesis that states a limit on what every small circuit can do. In each, the weight of the separation is carried by a derivation about what machines are and what they can do, or by a premise that states its result. Discharging the hypothesis, proving that factoring is not in BPP or that the language needs large circuits, brings back the derivation the hypothesis holds. That conditional separations are proved by ordinary means, with the derivation about machines held in their hypotheses, is the pattern the classifications lead one to expect.

14.3. What About Oracle Separations?

Question. Raz and Tal proved that BQP is not contained in the polynomial hierarchy relative to an oracle, a quantum advantage far stronger than BQP ≠ BPP. Baker, Gill and Solovay proved P ≠ NP relative to an oracle. Do these not refute the classifications?

Answer. They are proofs of relativized propositions, and the classifications concern the unrelativized separations. The relativized results are in fact the separating halves of the relativization barriers: because there are oracles on both sides, no relativizing argument settles the unrelativized question. Each is a formal, independent and sound proof of its own proposition, and each reasons, relative to its oracle, from what the machines are to what they can or cannot detect. Nothing in them bears against the statement that every proof of the unrelativized separation contains a step of the third kind.

14.4. Could a New Technique Evade All Three Barriers?

Question. The barriers name three families found between 1975 and 2009. A new technique could prove either separation while lying outside all three.

Answer. The question concerns the covering lemma (Lemma 8.6). By the lemma, a technique confined to the first two kinds of step relativizes, so a technique outside the relativizing family takes a step of the third kind. If such a technique proves a separation, the proof contains what the classification says it must contain, and it confirms the classification. Williams's method is the known example: it lies outside the three families, and its decisive step derives what can be decided about circuits of a class from what those circuits are. A new technique can evade all three barriers only by taking a step of the third kind of a form none of them blocks.

14.5. The Extended Church–Turing Thesis

Question. BQP $\neq$ BPP is a formal counterpart of the negation of the extended Church–Turing thesis, which says that every physically realizable model of computation is simulated by a probabilistic Turing machine with polynomial overhead. The classification of BQP $\neq$ BPP inherits whatever standing that thesis has.

Answer. The extended thesis is a claim about the relation between physical processes and formal computation, and I do not classify the extended thesis in this book. BQP $\neq$ BPP is a precise conjecture about two complexity classes, and its classification rests on the barrier results for those classes and on the premises of Chapter 10, not on any analysis of the thesis. The FIS book classifies a different thesis, the Church–Turing thesis, as Type T, by a demonstration that uses the definition of a formal system (FIS Section 10.8). That definition speaks of Turing machines and not of polynomial-time quantum or probabilistic ones, and I give no corresponding demonstration here for the extended thesis or for BQP $\neq$ BPP.

14.6. Are the Engagement Bounds Formal Theorems?

Question. The assignments of the three barrier families to kinds of step are not theorems in the sense the barrier results are. If they are informal, the classification lacks a formal foundation.

Answer. The assignments are classifications of each family's operations under the tests of Definition 6.2, checked operation by operation, and I state that standing in Section 8.3. The deduction of Theorem 8.9 does not use them. The deduction of Theorem 8.9 uses the relativization barrier, which says that no relativizing proof establishes the separation; the covering lemma, which says that every proof confined to the first two kinds relativizes; and Premise L. The engagement bounds say what each barrier shows about the kinds of step, and a counterexample to one of the engagement bounds would change that account without touching the classification.

14.7. What About Geometric Complexity Theory?

Question. Geometric complexity theory, the program of Mulmuley and Sohoni (2001), uses algebraic geometry and representation theory and is designed to evade all three barriers. Does the classification rule it out?

Answer. No. The program's main target is the permanent-versus-determinant question, the form of Valiant's algebraic counterpart of P versus NP (Valiant, 1979) that Mulmuley and Sohoni chose, which is generally regarded as more approachable than P versus NP and is not known to imply it. The program embeds the two polynomials in orbit closures under the action of a general linear group and seeks representation-theoretic obstructions to embedding one in the other. Bürgisser, Ikenmeyer and Panova (2019) showed that the simplest obstructions, which record only whether a representation occurs, do not suffice. A yes-or-no record of occurrence is a classification, a step of the second kind at most. The program's turn to multiplicities and positivity moves it toward reasoning from the algebraic structure of a computation's orbit to what the computation can do, which is the third kind. If it succeeds, the classification says what its proof will contain, and nothing in the classification says the proof would fail to be independent.

14.8. What About the Proved Separations at Smaller Scales?

Question. Bravyi, Gosset and König proved a separation of quantum from classical circuits, and Paul, Pippenger, Szemerédi and Trotter a separation of deterministic from nondeterministic time, without any assumption. The two proofs are formal, sound proofs that what one model is lets that model do what the other model cannot. What do they do to the framework?

Answer. The two proved separations at smaller scales are the clearest cases in the literature of what I claim and what I do not claim, and both are consistent with every result of this book. Each proof derives a capacity of one model and a limit of the other from what the two models are, as Chapters 9 and 11 find every proof of the larger separations must. Each is independent: its premises are the definitions of the models and of the problem, and none of them by itself implies the separation it proves. So each shows that a proof can derive what machines can do from what they are and still assume nothing about the answer. Neither proves the larger separation: one separates constant-depth circuit classes on a relation problem, and the other separates linear time on multitape machines. Each marks what a proof of the larger separation could look like: a derivation from what the two models are to what they can do, carried out from the definitions.

14.9. Does the Pseudorandomness Assumption Make the Classification of P ≠ NP Circular?

Question. The natural proofs barrier assumes strong pseudorandom function generators, which imply one-way functions, which imply P ≠ NP. So the classification of P ≠ NP rests on an assumption that already contains P ≠ NP.

Answer. The classification does not use the assumption: Theorem 12.1 rests on the relativization barrier, the covering lemma and Premise L (Section 12.5). The assumption enters only the natural proofs barrier, which I use in this book to say which steps of the third kind are not enough. Were it a premise of the classification, it would still not be a premise of any proof of P ≠ NP, since the classification states what such a proof contains and does not conclude that P ≠ NP holds.

14.10. Do the Classifications Say the Separations Cannot Be Proved?

Question. If every technique the barriers cover fails, and every proof must take a step that these techniques do not take, the classifications amount to a claim that the separations cannot be proved.

Answer. The classifications do not amount to a claim that the separations cannot be proved. A third-kind classification says what every proof must contain. A third-kind classification does not say that no proof exists, and does not say that every proof must assume part of the proof's conclusion. A proof that takes a step of the third kind about what the machines of the two models can and cannot do, from the definitions of the two models, is consistent with every result of this book, and it could be formal, independent and sound. The two proved separations at smaller scales are proofs of exactly that kind. The FIS Theorem would say more only for a separation shown to be inescapably circular, and I show neither separation to be inescapably circular.

14.11. Summary

The questions share a pattern. Each points to a feature of the classifications, their shared template, the existence of conditional, relativized or smaller-scale results, the standing of some of their parts, or the reach of their conclusion, and asks whether the feature undermines them. The shared template does not make one classification follow from the other. The conditional, relativized and smaller-scale results are proofs of other propositions, and those results reason in the way the classifications describe. The covering lemma is proved, and the engagement bounds are stated with their counterexamples. And the conclusion of each classification is a statement of what every proof must contain, not a statement that no proof exists. The classifications can be refuted, and the refutations are definite: an argument confined to the first two kinds that fails relative to some oracle, or a proof whose only steps of the third kind are about processes other than the machines the separation compares.

Chapter 15. What the Book Establishes

ABSTRACT

This chapter states what I establish in the book and collects the premises I use. In this book I establish the covering lemma: every argument confined to the first two kinds of step relativizes. I also establish a template that gives the third-kind classification of the separation of two complexity classes from three things: the relativization barrier for the pair of classes, the covering lemma, and a step premise. I apply the template to BQP ≠ BPP and to P ≠ NP: under the step premise, every formal, sound proof of either separation that does not assume that separation openly contains a step that derives what the machines of one model can do, or what the machines of the other cannot do, from what those machines are. The classifications say what every proof must contain. They do not apply the FIS Theorem to either separation.

15.1. What Is Established

The covering lemma (Lemma 8.6). Under the boundary test of Section 6.2, every argument about complexity classes whose steps and cited premises are all of the first two kinds relativizes; so a statement that fails relative to some oracle needs a step of the third kind in every proof (Corollary 8.7).

The template (Theorem 8.9). Let $D \subseteq C$ be classes defined by the polynomial-time machines of two models. If there is an oracle relative to which C is contained in D, then under the step premise $\mathrm{L}(C, D)$ the separation of C from D is a third-kind proposition.

The classification of BQP ≠ BPP (Theorem 10.1). Under Premise $\mathrm{L}(\mathrm{BQP}, \mathrm{BPP})$, every formal, sound proof of BQP ≠ BPP that assumes no aspect of it openly contains a step of the third kind that derives what quantum circuits do from what they are, or a limit on what probabilistic machines can do from what they are.

The classification of P ≠ NP (Theorem 12.1). Under Premise L(NP, P), every formal, sound proof of P ≠ NP that assumes no aspect of it openly contains a step of the third kind that derives what nondeterministic machines or verifiers do from what they are, or a limit on what deterministic polynomial-time machines, or the polynomial-size circuits that simulate them, can do from what they are.

15.2. The Premises

The classifications rest on three kinds of ground.

Proved results used in the deduction: the relativization barrier for each pair, of Baker, Gill and Solovay for P and NP and of a PSPACE-complete oracle for BQP and BPP (Theorems 4.3 and 4.4); the covering lemma (Lemma 8.6), resting on the boundary test of Section 6.2; and the sector-elimination lemma (Lemma 6.6), with the definitions it uses, imported from the FIS book.

One named premise, stated where it is used with its counterexample: Premise L for each separation (Sections 10.2 and 12.2), which says that the step of the third kind each proof then contains derives what the machines the separation compares do or cannot do.

Results and classifications that explain and are not used in the deduction: the natural proofs barrier of Razborov and Rudich, with the pseudorandomness assumption it rests on, and the theorem of Adleman, Bennett and Gill that BPP is contained in P/poly; the algebrization barrier of Aaronson and Wigderson; and the engagement bounds of the three families (Propositions 8.2 to 8.4).

Ground	Statement	Used by	Standing
(R), relativization barrier	An oracle makes the stronger class part of the weaker (Theorems 4.3, 4.4)	Theorems 8.9, 10.1, 12.1	Proved
Boundary test	A step reasoning from a process's inner makeup to what it does is of the third kind (Definition 6.2a)	Lemma 8.6, Propositions 8.2 to 8.4	Definition, checked on the cases of Section 6.2
Covering lemma	Every argument within the first two kinds relativizes (Lemma 8.6)	Theorems 8.9, 10.1, 12.1	Proved
Sector elimination	Lemma 6.6	Theorem 8.9	Proved (FIS book)
Premise L	The third-kind step is about the machines the separation compares	Theorems 8.9, 10.1, 12.1	Named premise
(N), natural proofs barrier	Theorem 4.7, under strong pseudorandom function generators	Proposition 8.3 (explanation)	Proved, under the assumption
(A), algebrization barrier	Theorem 4.10	Proposition 8.4 (explanation)	Proved

15.3. What the Results Imply

The results explain the barriers. Relativization eliminates every technique confined to the first two kinds of step, by the covering lemma; the natural proofs and algebrization barriers eliminate two forms of the third kind. What every proof of either separation needs, and what the eliminated families lack, is a step that reasons from what the machines of a model are to what they can or cannot do.

The results say where a proof of either separation must look. Refinements of relativizing, natural or algebrizing techniques will not reach it. A proof must contain a derivation, from what quantum circuits and probabilistic machines are, or from what nondeterministic and deterministic machines are, of

what they can and cannot do. Williams's algorithmic method, which derives circuit lower bounds from algorithms built on the gates of the circuits of a class, is a known technique that takes such a step.

The template applies to any separation of classes $D \subseteq C$ for which the relativization barrier holds, with its own step premise. For each such separation it gives the same statement of what every proof must contain; and by Corollary 8.7, every statement about complexity classes that fails relative to some oracle needs a step of the third kind in every proof.

15.4. The Scope of the Results

The classifications are statements of what every proof must do. They are not statements of what every proof must assume, and they do not apply the FIS Theorem. Neither separation is a claim about every member of a class to which every formal proof belongs, so the FIS book's self-dependence theorem does not apply to either, and I give no demonstration that every formal derivation of either separation assumes part of that separation. I therefore make no claim that either separation is inescapably circular, that either is Type T, or that either lacks a proof that is formal, independent and sound.

Two proved results show the difference. The separation of shallow quantum from shallow classical circuits (Bravyi, Gosset and König, 2018) and the separation of nondeterministic from deterministic linear time (Paul, Pippenger, Szemerédi and Trotter, 1983) each derive a capacity of one model and a limit of the other from what the two models are, and each is formal, independent and sound. Reasoning about what the machines a separation compares can and cannot do is not assuming the separation.

The results do not say whether either separation is true. The evidence of Chapter 3 bears on whether either separation is true, and I have not added to that evidence in this book.

15.5. Summary

I began this book with two separations of the same shape, each saying that the way a machine branches changes what it can do in polynomial time, and each blocked by the same three families of proof techniques. I asked what every proof of such a separation must contain.

The answer has a definite form. Every step of a formal proof is of one of three kinds, and every argument confined to the first two kinds relativizes, by the covering lemma. A short lemma turns the failure of relativizing techniques into the statement that every proof of the separation takes a step of the third kind; a step premise places that step among the derivations of what the machines the separation compares can or cannot do. One template gives this for any separation with the relativization barrier, and I apply the template to BQP $\neq$ BPP and to P $\neq$ NP. The natural proofs and algebrization barriers show which steps of the third kind are not enough. Under the step premise, every proof of either separation must derive what the machines of one model can do, or what the machines of the other cannot do, from what those machines are.

The answer concerns what proofs must do and not what they must assume. A proof that derives a separation from the definitions of the two models would contain the step the classification requires, and it could be formal, independent and sound. The premise is named, every ground has its counterexample, and every claim is exactly as strong as the argument for it.

APPENDICES

The Barrier Theorems and the Comparison Algebra

Appendix A. The Barrier Theorems: Proof Sketches

ABSTRACT

This appendix sketches the proofs of the barrier theorems of Chapter 4, for both pairs of classes, at the level of detail needed to see what each proof uses. The sketches follow the original papers; the reader who wants the full arguments will find them there.

A.1. Relativization

An oracle that makes every class equal. Let A be a language complete for PSPACE. A deterministic polynomial-time machine with access to A decides every language in PSPACE with one query, so PSPACE $\subseteq \mathrm{P}^A$. In the other direction, every machine of the four classes, nondeterministic, probabilistic or quantum, running in polynomial time with access to A, can be simulated in polynomial space: a polynomial-space machine answers each query to A itself; it tries every path of a nondeterministic machine in turn, reusing its space; it runs through every sequence of coin flips of a probabilistic machine in turn and counts the accepting ones; and it computes each amplitude of a quantum machine as a sum over the paths of the computation, one path at a time (Bernstein and Vazirani, 1997). So P^A, NP^A, BPP^A and BQP^A all equal PSPACE.

An oracle that separates P from NP. Build B in stages, and let L_B be the set of strings 1^n such that B contains some string of length n. L_B is in NP^B: guess a string of length n and check it with one query. At stage i, take the i-th deterministic oracle machine M_i with its polynomial time bound p_i, choose a length n larger than every length used so far with $2^n > p_i(n)$, and run M_i on 1^n, answering no to every query not yet decided. If it accepts, add no string of length n to B; if it rejects, add a string of length n that it did not query, which exists because it made fewer than 2^n queries. Either way M_i errs on 1^n, so no deterministic polynomial-time machine decides L_B relative to B.

An oracle that separates BQP from BPP. Simon's problem: an oracle gives a function f from n-bit strings to n-bit strings, which is either one-to-one or satisfies, for distinct x and y, $f(x) = f(y)$ exactly when x and y differ by a fixed nonzero string s. A quantum algorithm queries f on an equal superposition of all inputs, applies the Hadamard transform, and measures, obtaining a random string orthogonal to s; about n repetitions determine s by linear algebra. A classical algorithm learns nothing about s until it finds two inputs with the same value, and by the birthday bound it needs about $2^{n/2}$ queries to do so with fair probability. A diagonal construction over the probabilistic machines then gives an oracle B with $\mathrm{BQP}^B \not\subseteq \mathrm{BPP}^B$ (Simon, 1997; Bernstein and Vazirani, 1997, gave an earlier superpolynomial separation by recursive Fourier sampling).

What the proofs use. The proofs compare counts (2^n against $p_i(n)$, the birthday bound), enumerate machines, simulate, and branch on the answers of a machine at each stage. They are themselves formal, independent and sound proofs of their claims. The family they eliminate, the relativizing techniques, is a different thing from the proofs that eliminate it: the FIS book counts it as using its oracles only in steps of the first kind (Proposition 8.2).

A.2. Natural Proofs

A pseudorandom function generator takes a random seed of length k and yields a Boolean function, on inputs of some length n, that a small circuit computes from the seed, and that no circuit of size 2^{k^ε} can tell apart from a uniformly random function when it may query the function. Goldreich, Goldwasser and Micali (1986) built such generators from pseudorandom generators, which follow from one-way functions.

Suppose a natural proof used a property C_n. By usefulness, no function with small circuits has C_n, so no function the generator yields has it. By largeness, a random function has C_n with probability at least $2^{-\mathrm{O}(n)}$. By constructivity, C_n can be tested in time $2^{\mathrm{O}(n)}$ by reading the whole truth table, which takes 2^n queries. So testing C_n tells the generator's functions apart from

random ones. Choosing n to be a small enough power of k, the test runs in time below 2^{k^ε}, which contradicts the security of the generator. So no such property exists (Razborov and Rudich, 1997).

The argument uses counting, a probability bound, a test of a truth table and a contradiction. The family it eliminates is the family of proofs that rest on a cheap, common property of truth tables, which, by Proposition 8.3, take a step of the third kind in proving that the property is useful and use its result only through the property.

A.3. Algebrization

A pair that makes the classes equal. Let A be complete for PSPACE and $\tilde{A}$ its multilinear extension over finite fields. The multilinear extension of a language in PSPACE can be computed in polynomial space (Babai, Fortnow and Lund, 1991), so a nondeterministic, probabilistic or quantum polynomial-time machine with access to $\tilde{A}$ can be simulated in polynomial space, as in Section A.1. Polynomial space is contained in P^A. So $\mathrm{NP}^{\tilde{A}} \subseteq \mathrm{P}^A$ and $\mathrm{BQP}^{\tilde{A}} \subseteq \mathrm{BPP}^A$. An algebrizing proof of NP $\not\subseteq$ P would show $\mathrm{NP}^{\tilde{O}} \not\subseteq \mathrm{P}^O$ for every pair, and the pair of A and $\tilde{A}$ is one for which $\mathrm{NP}^{\tilde{A}} \not\subseteq \mathrm{P}^A$ fails. The same holds for BQP and BPP.

Pairs that separate the classes. Aaronson and Wigderson construct oracles B with extensions $\tilde{B}$ such that $\mathrm{NP}^B \not\subseteq \mathrm{P}^{\tilde{B}}$ (their Theorem 5.3), such that $\mathrm{NP}^B \not\subseteq \mathrm{BPP}^{\tilde{B}}$ (their Theorem 5.4, by a direct construction, and again in their Theorem 5.11, by communication complexity), and such that $\mathrm{BQP}^B \not\subseteq \mathrm{BPP}^{\tilde{B}}$ (their Theorem 5.11). Their Theorem 5.11 transfers lower bounds on communication complexity into lower bounds against machines that query a low-degree extension: a machine that queries the extension of an oracle split between two parties can be simulated by a communication protocol between them. For NP against BPP, by this route, they use the randomized communication lower bound for disjointness of Kalyanasundaram and Schnitger (1992) and of Razborov (1992); for BQP against BPP they use the exponential separation of quantum from randomized communication complexity due to Raz (1999).

What the proofs use. The proofs use field arithmetic, simulation in polynomial space, and communication lower bounds. The family they eliminate, the algebrizing techniques, takes a step of the third kind through the completeness results it cites and uses it only in arithmetized form (Proposition 8.4). Aaronson and Wigderson showed that the proofs of IP = PSPACE and MIP = NEXP algebrize (their Theorems 3.7 and 3.8).

A.4. Summary

Each barrier is proved by a short construction: a PSPACE-complete oracle that collapses the classes, a stage-by-stage or query-complexity construction that separates them, a contradiction with the security of pseudorandom generators, and the same oracle with its multilinear extension in the algebrized setting. Each proof is formal, independent and sound. The families they eliminate are what the template of Chapter 8 uses.

Appendix B. The Three Kinds as the Sectors of the Comparison Algebra

ABSTRACT

The FIS book shows that the three kinds of step are the three sectors of an algebra derived from the structure of comparison, and that the three primitives of computation are the same three sectors. This appendix states that result as the FIS book gives it. It shows where the kinds come from. The definitions, theorems and premises of this book are stated and proved without the result of this appendix.

B.1. The Algebra and Its Sectors

This appendix is for readers of the FIS book and its companion work on noncommutative geometry; I do not use the appendix in the rest of the book.

The FIS book derives, from the structural requirements of comparison together with the postulates of its companion work on noncommutative geometry, the algebra

$$A_F = \mathbb{C} \oplus \mathbb{H} \oplus M_3(\mathbb{C}),$$

the direct sum of the complex numbers, the quaternions and the 3×3 complex matrices (FIS Chapter 4 and Appendix A). Its three central idempotents e_1, e_2 and e_3 (elements that commute with the whole algebra, each equal to its own square) project onto the three summands, and the labels of the three kinds of step are their names. The summands differ in how far they commute: $\mathbb{C}$ is commutative; $\mathbb{H}$ is noncommutative and a division algebra, so that exchanging two factors changes the product, as $ij = k$ and $ji = -k$; and $M_3(\mathbb{C})$ is noncommutative and not a division algebra, with a fundamental representation of dimension three.

The modal binding of the FIS book binds each summand to one of the three pairs of comparison: sameness and difference to $\mathbb{C}$, whose comparisons do not depend on the order of their terms; the subjective and the objective to

$\mathbb{H}$, whose comparisons have two roles that cannot be exchanged; and content and context, taken for a process as what it is and what it does, to $M_3(\mathbb{C})$. The binding is a structural equivalence with its inclusions shown: the defining character of each summand falls within the definition of the pair it is bound to, and the form, three summands to three pairs, one to one, is the form of the three roles of comparison (FIS Section 4.10). The three kinds of step of Definition 6.1 are the three kinds of comparison so bound.

B.2. The Primitives of Computation

The FIS book also shows that the three primitives of computation are the three sectors: sequentiality, the chaining of dependent steps, with e_1; conditionality, branching on examined data, with e_2; and cyclicity, the revisiting of states, which lets a short program run for a long time, with e_3 (FIS Chapter 6). Removing each primitive gives a characteristic weaker device: without sequentiality a combinational circuit, without conditionality a straight-line program, without cyclicity a decision tree.

So the two separations of this book are differences in the mode of conditionality: existential against deterministic branching for P and NP, interference of amplitudes against independent coin flips for BQP and BPP. The mode of conditionality is what the machines of each pair differ in. What the classifications of Chapters 10 and 12 say a proof needs is a different thing: a step of reasoning, of the third kind, that derives what machines can do from what they are. The difference in the mode of conditionality is a fact about the subject of a separation; the need for a step of the third kind is a fact about the proofs of that separation.

B.3. The Kind of Result

The correspondence of primitives and sectors is a structural equivalence: the character of each primitive falls within the character of its sector, and the form of the three primitives, with the weaker device left when each is removed, falls within the form of the three sectors. A structural equivalence with its inclusions shown establishes its result as a theorem does (FIS Section

6.5). The definitions of the kinds of step are assigned by the tests of the engagement template directly, and the FIS Theorem, the sector-elimination lemma and every result of this book are proved without the correspondence.

B.4. Summary

The FIS book shows that the three kinds of step are the three sectors of the comparison algebra, bound to sameness and difference, to the subjective and the objective, and to what a process is and what it does. The primitives of computation are the same three sectors. These results show where the kinds come from, and the results of this book are proved without them.

Glossary

Algebrizing technique. A technique whose argument still holds after an oracle is paired with a low-degree extension of it, the extension given to the class that must be shown the stronger (Definition 4.9).

Aspect of a proposition. A claim the proposition asserts, whole or in part (Definition 7.1).

Assuming a claim openly. One premise the argument does not establish, by itself, implies the claim (Definition 7.2).

Assuming a claim silently. The statement that the argument meets a condition it must meet to count as a formal or sound proof, by itself, implies the claim (Definition 7.2).

Assuming a claim through what an argument rests on. The argument rests on a premise or rule it does not establish, every derivation of which assumes the claim (Definition 7.2).

Boundary test. A step that uses a process only as a computation is of the first kind; a step whose object is the inner makeup of a process and what that makeup makes it do is of the third (Definition 6.2a).

Computation, in the defined sense. A sequence of configurations, each obtained from the one before by one application of the process's step rule, with the resources the sequence uses (Definition 6.2a).

Covering lemma. Every argument whose steps and cited premises are all of the first two kinds relativizes (Lemma 8.6).

Criteria 1 to 5. The five criteria of the FIS book's procedure for a proposition, listed in Section 8.2.

Engagement. The set of kinds of step a proof uses, in its own steps and in the premises it cites (Definition 6.1).

Engagement bound. A class of proofs is E-bounded when no proof in it uses a kind of step outside E (Definition 6.4).

First kind of step. A step whose verdict does not depend on the order of its terms: identity, substitution, applying a definition, counting, enumerating, building a definite object, and running a process (Definitions 6.1 and 6.2a).

Hypothesis E. Every proof of a separation within the first two kinds relativizes, algebrizes, or is natural; it follows from the covering lemma (Corollary 8.8).

Inescapably circular. Every formal derivation of the proposition assumes some aspect of it (Definition 7.3).

Inner makeup. The particular form of a process's step rule, the kinds of its gates, or the body of its loop (Definition 6.2a).

Local checkability. A computation is correct exactly when every small window of its tableau is correct; a fact about the inner makeup of Turing machine steps (Section 8.5).

Natural proof. A proof of a circuit lower bound through a property of truth tables that is cheap to test, held by many functions, and held by no function with small circuits (Definition 4.6).

Operative sense. A proposition requires the third kind when every proof of it in the class, should one exist, counts the third kind (Section 6.3).

Premise L, the step premise. The step of the third kind each proof of a separation contains derives what the machines the separation compares do or cannot do (Section 8.5).

Relativizing technique. A technique whose argument goes through unchanged if every machine is given the same arbitrary oracle (Definition 4.2).

Required engagement. The kinds of step every formal, sound proof of a proposition uses, among the proofs that do not assume it openly (Definition 6.3).

Second kind of step. A step relating two terms in roles that cannot be exchanged: case analysis with different arguments, branching, a query and its answer, a challenge and its response (Definition 6.1).

Sector-elimination lemma. If technique families that contain every proof within the first two kinds each fail to establish a proposition, every proof of it uses the third kind (Lemma 6.6).

Separation. The proposition that some language of the stronger class is not in the weaker (Definition 8.1).

Third kind of step. A step that derives what a process does from what it is, or a limit on what a class of processes can do from what its members are (Definitions 6.1 and 6.2a).

Third-kind proposition. Every formal, sound proof of it that does not assume it openly uses a step of the third kind about the processes it is about (Definition 6.5).

Type S, Type P, Type T. Required engagement $\{e_1\}$; required engagement $\{e_1, e_2\}$; a third-kind proposition that is inescapably circular (Definitions 6.5 and 7.4).

Valid proof. A proof that is at once formal, independent and sound (Section 7.1).

Bibliography

Aaronson, S. (2010). "BQP and the Polynomial Hierarchy". *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, 141–150.

Aaronson, S. and Ambainis, A. (2015). "Forrelation: A Problem that Optimally Separates Quantum from Classical Computing". *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, 307–316.

Aaronson, S. and Arkhipov, A. (2011). "The Computational Complexity of Linear Optics". *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, 333–342.

Aaronson, S. and Wigderson, A. (2009). "Algebrization: A New Barrier in Complexity Theory". *ACM Transactions on Computation Theory*, 1(1), Article 2.

Adleman, L. (1978). "Two Theorems on Random Polynomial Time". *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, 75–83.

Ajtai, M. (1983). "Σ^1_1-Formulae on Finite Structures". *Annals of Pure and Applied Logic*, 24(1), 1–48.

Arora, S., Impagliazzo, R., and Vazirani, U. (1992). "Relativizing versus Nonrelativizing Techniques: The Role of Local Checkability". Manuscript.

Arute, F. and others (2019). "Quantum Supremacy Using a Programmable Superconducting Processor". *Nature*, 574, 505–510.

Aydinlioglu, B. and Bach, E. (2016). "Affine Relativization: Unifying the Algebrization and Relativization Barriers". Electronic Colloquium on Computational Complexity, Report TR16-040 (revised journal version).

Babai, L., Fortnow, L., and Lund, C. (1991). "Non-Deterministic Exponential Time Has Two-Prover Interactive Protocols". *Computational Complexity*, 1(1), 3–40.

Baker, T., Gill, J., and Solovay, R. (1975). "Relativizations of the P =? NP Question". *SIAM Journal on Computing*, 4(4), 431–442.

Beigel, R. and Tarui, J. (1994). "On ACC". *Computational Complexity*, 4(4), 350–366.

Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. (1997). "Strengths and Weaknesses of Quantum Computing". *SIAM Journal on Computing*, 26(5), 1510–1523.

Bennett, C. H. and Gill, J. (1981). "Relative to a Random Oracle A, $P^A \neq NP^A \neq$ co-NP^A with Probability 1". *SIAM Journal on Computing*, 10(1), 96–113.

Bernstein, E. and Vazirani, U. (1997). "Quantum Complexity Theory". *SIAM Journal on Computing*, 26(5), 1411–1473.

Bravyi, S., Gosset, D., and König, R. (2018). "Quantum Advantage with Shallow Circuits". *Science*, 362(6412), 308–311.

Bremner, M. J., Jozsa, R., and Shepherd, D. J. (2011). "Classical Simulation of Commuting Quantum Computations Implies Collapse of the Polynomial Hierarchy". *Proceedings of the Royal Society A*, 467(2126), 459–472.

Bürgisser, P., Ikenmeyer, C., and Panova, G. (2019). "No Occurrence Obstructions in Geometric Complexity Theory". *Journal of the American Mathematical Society*, 32(1), 163–193.

Chen, L., Hirahara, S., and Ren, H. (2024). "Symmetric Exponential Time Requires Near-Maximum Circuit Size". *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, 1990–1999.

Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., and Spielman, D. A. (2003). "Exponential Algorithmic Speedup by a Quantum Walk". *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 59–68.

Cook, S. A. (1971). "The Complexity of Theorem-Proving Procedures". *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 151–158.

Feynman, R. P. (1982). "Simulating Physics with Computers". *International Journal of Theoretical Physics*, 21(6/7), 467–488.

Fortnow, L. and Sipser, M. (1988). "Are There Interactive Protocols for co-NP Languages?". *Information Processing Letters*, 28(5), 249–251.

Furst, M., Saxe, J. B., and Sipser, M. (1984). "Parity, Circuits, and the Polynomial-Time Hierarchy". *Mathematical Systems Theory*, 17(1), 13–27.

Goldreich, O., Goldwasser, S., and Micali, S. (1986). "How to Construct Random Functions". *Journal of the ACM*, 33(4), 792–807.

Grover, L. K. (1996). "A Fast Quantum Mechanical Algorithm for Database Search". *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212–219.

Hartmanis, J. and Stearns, R. E. (1965). "On the Computational Complexity of Algorithms". *Transactions of the American Mathematical Society*, 117, 285–306.

Håstad, J. (1987). *Computational Limitations of Small-Depth Circuits*. MIT Press.

Impagliazzo, R., Kabanets, V., and Kolokolova, A. (2009). "An Axiomatic Approach to Algebrization". *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 695–704.

Impagliazzo, R. and Wigderson, A. (1997). "P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma". *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 220–229.

Kalyanasundaram, B. and Schnitger, G. (1992). "The Probabilistic Communication Complexity of Set Intersection". *SIAM Journal on Discrete Mathematics*, 5(4), 545–557.

Karp, R. M. (1972). "Reducibility Among Combinatorial Problems". In R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum, 85–103.

Kitaev, A. Yu. (1997). "Quantum Computations: Algorithms and Error Correction". *Russian Mathematical Surveys*, 52(6), 1191–1249.

Ladner, R. E. (1975). "On the Structure of Polynomial Time Reducibility". *Journal of the ACM*, 22(1), 155–171.

Landry, F. (2026). *Formalizability, Independence, Soundness; The FIS Theorem and the Three Limits of Proof*. Wytheville, Virginia: UVSM Publications.

Landry, F. (2026). *Non-Commutative Geometry and the Immanent Metaphysics*. Wytheville, Virginia: UVSM Publications.

Lautemann, C. (1983). "BPP and the Polynomial Hierarchy". *Information Processing Letters*, 17(4), 215–217.

Levin, L. A. (1973). "Universal Sequential Search Problems". *Problems of Information Transmission*, 9(3), 265–266.

Li, Z. (2024). "Symmetric Exponential Time Requires Near-Maximum Circuit Size: Simplified, Truly Uniform". *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, 2000–2007.

Lloyd, S. (1996). "Universal Quantum Simulators". *Science*, 273(5278), 1073–1078.

Lund, C., Fortnow, L., Karloff, H., and Nisan, N. (1992). "Algebraic Methods for Interactive Proof Systems". *Journal of the ACM*, 39(4), 859–868.

Madsen, L. S. and others (2022). "Quantum Computational Advantage with a Programmable Photonic Processor". *Nature*, 606, 75–81.

Mulmuley, K. D. and Sohoni, M. (2001). "Geometric Complexity Theory I: An Approach to the P vs. NP and Related Problems". *SIAM Journal on Computing*, 31(2), 496–526.

Murray, C. and Williams, R. (2018). "Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP". *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, 890–901.

Paul, W. J., Pippenger, N., Szemerédi, E., and Trotter, W. T. (1983). "On Determinism versus Non-Determinism and Related Problems". *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, 429–438.

Raz, R. (1999). “Exponential Separation of Quantum and Classical Communication Complexity”. *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, 358–367.

Raz, R. and Tal, A. (2019). “Oracle Separation of BQP and PH”. *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, 13–23.

Razborov, A. A. (1985). “Lower Bounds on the Monotone Complexity of Some Boolean Functions”. *Soviet Mathematics Doklady*, 31, 354–357.

Razborov, A. A. (1987). “Lower Bounds on the Size of Bounded Depth Circuits over a Complete Basis with Logical Addition”. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4), 333–338.

Razborov, A. A. (1992). “On the Distributional Complexity of Disjointness”. *Theoretical Computer Science*, 106(2), 385–390.

Razborov, A. A. and Rudich, S. (1997). “Natural Proofs”. *Journal of Computer and System Sciences*, 55(1), 24–35.

Shamir, A. (1992). “IP = PSPACE”. *Journal of the ACM*, 39(4), 869–877.

Shor, P. W. (1994). “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 124–134.

Simon, D. R. (1997). “On the Power of Quantum Computation”. *SIAM Journal on Computing*, 26(5), 1474–1483.

Sipser, M. (1983). “A Complexity Theoretic Approach to Randomness”. *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 330–335.

Smolensky, R. (1987). “Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity”. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, 77–82.

Stearns, R. E., Hartmanis, J., and Lewis, P. M. (1965). “Hierarchies of Memory Limited Computations”. *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, 179–190.

Turing, A. M. (1936). “On Computable Numbers, with an Application to the Entscheidungsproblem”. *Proceedings of the London Mathematical Society*, Series 2, 42, 230–265.

Valiant, L. G. (1979). “Completeness Classes in Algebra”. *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, 249–261.

Williams, R. (2014). “Nonuniform ACC Circuit Lower Bounds”. *Journal of the ACM*, 61(1), Article 2.

Zhong, H.-S. and others (2020). “Quantum Computational Advantage Using Photons”. *Science*, 370(6523), 1460–1463.

Index

A

Adleman's theorem, 2.5, 8.1, 10.1, 15.2

algebrization barrier, 1.2, 4.4, 4.5, 4.6, 5.1, 8.4, 8.6, 8.7, 10.3, 10.4, 10.5, A.3

amplitude, 1.1, 2.4, 2.7, 2.8, 3.2, 9.1, 9.2, 9.4, 9.6, 10.1, A.1, B.2

assuming a claim, openly, silently, or through what an argument rests on, 7.2, 6.3

aspect of a proposition, 6.3, 6.5, 6.6, 7.1, 7.2, 7.3, 8.4, 8.5, 8.6, 10.2, 12.2, 15.1

B

Baker, Gill and Solovay theorem, 1.2, 4.2, 12.1, 14.3, 15.2

barrier theorems, 1.2, 1.3, 3.1, 4.4, 4.5, 5.3, 8.3, 8.4, 8.6, 10.1, 14.6, 15.3

boundary test, 6.2, 8.3, 8.5

boson sampling, 3.3

BPP, 1.1, 2.5, 2.7, 3.4, 4.2, 4.4, 8.1, 10.1, 10.2, 10.3, 10.7, A.3

BQP, 1.1, 2.7, 3.4, 4.2, 4.4, 10.1, 10.2, 10.3, 10.7, 13.3, 14.5, 15.1

Bravyi, Gosset and König separation, 13.3, 14.8, 15.4

C

Criteria 1 to 5, 8.2, 7.3, 13.1

Church–Turing thesis, 7.3, 14.5

circuit lower bounds, 1.2, 1.5, 2.5, 4.3, 4.5, 4.6, 4.7, 8.1, 8.7, 10.4, 13.4, A.3

conditional separations, 3.4, 3.5, 3.6, 13.5, 14.2

Cook–Levin theorem, 11.1, 13.4

covering lemma (Hypothesis E), 8.5, 1.4, 8.3, 8.6, 10.2, 10.3, 10.4, 10.7, 12.3, 12.4, 12.7, 14.4, 15.1

D

diagonalization, 1.2, 4.5, 8.3, 12.6

E

engagement, 5.3, 5.4, 6.1, 6.3, 6.4, 6.6, 6.7, 8.3, 8.5, 8.6, 8.7, 10.4

engagement bounds, 5.3, 5.4, 6.4, 6.6, 8.3, 8.7, 8.9, 14.6, 14.11, 15.2

extended Church–Turing thesis, 14.5

F

first kind of step, 1.4, 6.1, 6.2, 6.5, 6.7, 8.2, 8.3, 9.2, 11.2, A.1

FIS Theorem, 7.2, 7.3, 7.5, 8.8, 13.1, 13.5, 13.6, 14.10, 15.4, B.3

formal derivation, 1.3, 3.5, 7.1, 7.2, 7.3, 7.5, 8.8, 13.1, 13.5, 15.4

G

geometric complexity theory, 14.7

Grover's algorithm, 3.2

I

inescapably circular, 7.2, 7.3, 7.4, 7.5, 13.1, 14.10, 15.4

interference, 1.1, 2.4, 3.2, 4.2, 9.3, 9.4, 13.3, B.2

IP = PSPACE, 4.4, 10.4, A.3

L

linear time, deterministic and nondeterministic, 11.4, 11.5, 12.6, 13.4, 13.6, 14.8, 15.4

local checkability, 8.3, 8.5

low-degree extension, 4.4, 4.6, 4.7, 8.3, 8.4, 10.1, 12.1, 15.1, A.3, A.4

N

natural proofs barrier, 1.2, 1.4, 4.3, 4.5, 4.6, 4.7, 5.1, 8.3, 8.4, 8.7, 10.1, 10.5

NP, 2.3, 2.7, 3.1, 4.2, 4.4, 11.2, 12.1, 12.3, 12.4, 12.5, 12.7, 14.9

NP-completeness, 2.3, 2.7, 3.1, 3.6, 13.4, 13.5

O

one-way functions, 3.1, 4.3, 10.5, 12.5, 14.9, A.2

oracle, 1.2, 3.4, 4.1, 4.2, 4.4, 4.6, 5.1, 8.3, 8.4, 13.3, 14.3, A.1

P

P/poly, 2.5, 8.1, 8.4, 8.6, 10.1, 10.3, 11.3, 12.1, 12.2, 12.3, 15.1, 15.2

operative sense, 6.3

polynomial hierarchy, 2.6, 2.7, 3.1, 3.3, 3.4, 12.4, 13.3, 13.4, 14.3

Premise L (step premise), 1.4, 8.5, 8.6, 8.7, 8.9, 10.2, 10.3, 10.7, 12.2, 12.3, 12.7, 15.1

pseudorandom function generators, 1.2, 4.3, 4.5, 4.7, 5.1, 10.3, 10.7, 12.3, 12.5, 12.7, 15.1, A.2

Q

quantum circuit, 2.4, 2.7, 3.3, 5.2, 9.1, 9.2, 9.3, 9.4, 9.6, 10.2, 10.3, 13.3

R

Raz and Tal theorem, 3.4, 13.3, 14.3

reasoning about a relation, 1.3, 7.3

relativization barrier, 1.2, 4.1, 4.2, 4.4, 4.5, 4.6, 8.4, 8.6, 10.3, 10.4, 12.4, 14.3

required engagement, 6.3, 6.5, 6.6, 6.7

S

SAT, 2.3, 11.1, 11.2, 11.3, 13.4

second kind of step, 1.4, 6.1, 6.2, 6.5, 6.7, 8.3, 14.7

sector-elimination lemma, 5.4, 6.6, 6.7, 8.3, 8.5, 8.9, 15.2, B.3

self-dependent proposition, 7.4, 9.1, 11.1, 15.4

separation template, 1.4, 5.3, 6.2, 7.5, 8.3, 8.7, 8.9, 10.4, 10.7, 12.3, 14.1, 14.11

Shor's algorithm, 2.7, 3.2, 3.4, 3.6, 9.3, 13.3, 14.2

Simon's problem, 1.2, 3.4, 4.2, 9.3, 10.1, 13.3, 15.2, A.1

T

third kind of step, 6.1, 6.3, 6.5, 6.6, 6.7, 8.5, 8.6, 8.7, 10.4, 10.6, 10.7, 12.4

third-kind proposition, 1.4, 6.5, 6.6, 6.7, 7.2, 7.3, 8.6, 8.9, 10.3, 10.7, 12.3, 12.7

time hierarchy theorem, 4.1, 8.2, 9.2, 10.4, 11.2

Type S and Type P, 6.5, 9.5

Type T, 7.2, 7.3, 7.4, 14.5, 15.4

U

universal machine, 6.2, 8.5

V

valid proof, 7.1, 7.5

W

Williams's algorithmic method, 1.5, 10.4, 10.6, 10.7, 12.4, 13.4, 14.4, 15.3