Bernoulli Types

An Algebra of Approximate Computation

Alexander Towell

June 8, 2026

ORCID: `0000-0001-6443-9897`
Web: `https://metafunctor.com`
Source: `https://github.com/queelius/bernoulli-book`

Contents

Preface

This is a placeholder preface. The final preface will:

- State the book's thesis and audience.
- Declare prerequisites: undergraduate probability, basic discrete math, comfort with proof, working Python knowledge for the notebook companion.
- Explain the lockstep notebook discipline (some chapters carry notebooks; theory-heavy chapters do not, by design).
- Provide a reading guide for the smart-undergrad-upward reader (chapters that admit a soft landing) and the adjacent-field grad student (the modal reader).

Part I

The Atom

Chapter 1

The Noisy Bit

1.1 Three Faces of a Noisy Bit

Three systems from different corners of applied probability each produce a single Boolean output that may be wrong, and at the level of input-output behavior they are one object. We introduce them as running examples and extract that object.

The biased coin

Consider a coin whose bias depends on how it is held: palmed "in" it lands heads with probability τ, palmed "out" with probability ϵ, where τ and ϵ are neither equal nor near $1/2$. Read "in" as a positive condition and "out" as a negative one. The coin reports each with its own reliability: two inputs, one Boolean output, two independent parameters. It returns as an asymmetric channel in Section 2.2 and as the worked instance of the formal definition in Chapter 4.

The binary symmetric channel

A single bit crosses a noisy line, a wire, a wireless link, a chip-to-chip trace. For each bit independently the line flips the symbol with crossover probability ε and passes it through otherwise, with the same ε in both directions: a $\top$ flips to $\bot$ and a $\bot$ to $\top$ with equal probability. This is the binary symmetric channel (BSC), the simplest model of a noisy one-bit line and the most studied channel in information theory; the term *channel* is Shannon's, for any single-input, single-output stochastic transmission. The symmetry is a real constraint, since many physical channels are asymmetric and need a

separate rate per direction (Section 2.2). Chapter 2 gives the BSC a formal definition and a matrix representation (Section 2.1).

The classifier verdict

A spam filter emits a verdict on each message, spam ($\top$) or ham ($\bot$). It flags ham as spam at the false-positive rate ε and misses spam at the false-negative rate ω, and these are generally unequal: a filter that drops legitimate mail at 1% is a different product from one that leaks 1% of spam. Here the two parameters are forced by the problem rather than by happenstance, since the two errors are distinct harms with distinct costs. A score-thresholded classifier traces a curve of (ε, ω) pairs as its threshold sweeps, the trade-off the ROC curve re-axes; no threshold drives both rates to zero at once.

The shared object

A coin held in or out, a wire carrying one bit, a filter emitting one verdict: physical, informational, and algorithmic, yet identical in input-output behavior. Each maps a Boolean input to a Boolean output that is correct with some probability and wrong otherwise, and the error behavior is fixed by at most two numbers, the rate of reporting $\top$ on a $\bot$-input and the rate of reporting $\bot$ on a $\top$-input. The BSC carries one parameter because its two rates are constrained equal; the coin and the filter carry two. Two numbers describe everything probabilistic at the single-query level. We call this object a *noisy bit*.

The three faces used three vocabularies because each is natural in its own letters, and they reconcile directly: the coin's τ is the true-positive rate $\tau = 1-\omega$ and its ϵ is ε; the classifier's (ε, ω) are the same quantities; the BSC is the special case $\varepsilon = \omega$. Three commitments follow and hold throughout Part I. The noisy bit is *atomic*: one Boolean input slot, one Boolean output slot, nothing smaller, and everything larger (multi-bit channels, repeated-trial algorithms such as Miller-Rabin, classifier ensembles) is built from noisy bits, as chapters 2 and 4 show. It is *behavioral*: the description fixes the rates and says nothing about mechanism, so a biased coin, a thermal photon detector, and a logistic-regression classifier with the same rates are the same noisy bit. And it is *parsimonious*: at most two numbers per noisy bit. The rest of Part I is the algebra of how these atoms compose.

1.2 Quantifying the Noise

A noisy bit is fixed by two conditional error rates. We make "rate" precise through the 2×2 contingency table, name the four rates it yields, and relate them to the precision and recall of binary classification.

The contingency table

Run a noisy bit n times. Each trial has a known true input, $\top$ or $\bot$, and an observed output, so the n trials are ordered pairs (truth, output), each falling into one of four joint states: a *true positive* (truth $\top$, output $\top$), a *false negative* (truth $\top$, output $\bot$), a *false positive* (truth $\bot$, output $\top$), and a *true negative* (truth $\bot$, output $\bot$). The term "false negative" names the output, not the truth: a $\bot$ reported when the truth was $\top$. Writing TP, FN, FP, TN for the counts, so that $\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN} = n$, arrange them by truth (rows) and observed output (columns):

	output $\top$	**output** $\bot$	total
truth $\top$	TP	FN	TP + FN
truth $\bot$	FP	TN	FP + TN
total	TP + FP	FN + TN	n

Table 1.1: The 2×2 contingency table for n observations of a noisy bit. Rows condition on truth; the diagonal cells record agreement and the off-diagonal cells disagreement.

Rates

The two error rates are conditional, each normalized by its truth row:

$$\varepsilon \;=\; \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}, \qquad \omega \;=\; \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}.$$

Dividing by the row total rather than by n is what makes ε and ω properties of the noisy bit itself rather than of the input mixture observed. The complementary rates are the true-negative rate $\eta = 1 - \varepsilon$ (the $\bot$-inputs reported correctly, also *specificity*) and the true-positive rate $\tau = 1 - \omega$ (also *sensitivity*, and *recall* in the vocabulary below). The two free parameters of a noisy bit are ε and ω; the other two are derived. The biased coin's (τ, ϵ) of Section 1.1 is (τ, ε), and the binary symmetric channel of Section 2.1 is the constrained case $\varepsilon = \omega$, where one parameter suffices. These four symbols keep these meanings throughout the book.

Remark (Negation reflects the rate pair)**.** *Relabeling the output of a noisy bit, reporting $\perp$ wherever it reported $\top$ and conversely, exchanges the two error directions: a noisy bit with rates (ε, ω) becomes one with rates (ω, ε). Logical negation thus acts on the parameter pair as the reflection $(\varepsilon, \omega) \mapsto (\omega, \varepsilon)$ across the diagonal of the rate square $[0,1]^2$. This is the atom-level form of the complement operation that Part II lifts to approximate sets (Proposition 6.2.2).* $\triangle$

Precision and recall

A practitioner of binary classification reads Table 1.1 through precision and recall:

$$\text{precision} \;=\; \frac{\text{TP}}{\text{TP}+\text{FP}}, \qquad \text{recall} \;=\; \frac{\text{TP}}{\text{TP}+\text{FN}}.$$

Recall coincides with the true-positive rate, recall $= \tau = 1 - \omega$. Precision does not reduce to the rates of the noisy bit: its denominator is a column total, which depends on how many $\top$- and $\perp$-inputs the sample contained. With $n_\top$ positives and $n_\perp$ negatives, $\text{TP} \approx n_\top \tau$ and $\text{FP} \approx n_\perp \varepsilon$, so

$$\text{precision} \;\approx\; \frac{n_\top(1-\omega)}{n_\top(1-\omega) + n_\perp \varepsilon}.$$

Precision is therefore a function of the *prevalence* $n_\top / n_\perp$ as well as the rates, whereas (ε, ω) is intrinsic to the noisy bit. The pair (ε, ω) describes the noisy bit; the pair (precision, recall) describes the noisy bit together with a workload. This is why the framework takes (ε, ω) as the primary description and treats precision and accuracy as derived; Chapter 7 develops the derived measures as random variables.

An instance

A filter run on $n = 100$ messages, 40 spam ($\top$) and 60 ham ($\perp$), produces the table

	flagged spam	**flagged ham**	total
true spam	35	5	40
true ham	3	57	60
total	38	62	100

with rates $\varepsilon = 3/60 = 0.05$ and $\omega = 5/40 = 0.125$, hence $\eta = 0.95$ and $\tau = 0.875$. Recall is $35/40 = 0.875 = \tau$, while precision is $35/38 \approx 0.921$,

exceeding τ because the column total (38) falls short of the truth row total (40). At a different prevalence the same (ε, ω) would yield a different precision.

Remark (The constant noisy bit and the zero-information line)**.** *A noisy bit that ignores its input, reporting $\top$ with a fixed probability p regardless of the truth, has $\varepsilon = p$ and $\omega = 1-p$, so $\varepsilon+\omega = 1$. On this anti-diagonal of the rate square the output is statistically independent of the truth: the report carries no information about the input. A constant rule is thus simultaneously a perfect classifier for one truth class and the worst possible one for the other (at $p = 1$, $\omega = 0$ but $\varepsilon = 1$). The locus $\varepsilon+\omega = 1$ is the zero-information case, and it is distinct from the symmetric case $\varepsilon = \omega$, which remains informative for $\varepsilon \neq 1/2$.* △

1.3 A First Look at the Boolean Universe

A noisy bit has two inputs and two outputs, so its single-trial behavior is exhausted by four input-output combinations. Section 1.2 arranged these as a table of counts; abstracted from any particular run they are a table of outcome types, indexed by true input (rows) and observed report (columns):

	report $\top$	**report** $\bot$
truth $\top$	correct positive	false negative
truth $\bot$	false positive	correct negative

The four cells carry the same labels as the contingency table of Section 1.2, TP, FN, FP, TN, seen from a different angle: the contingency table counts how many of n trials land in each cell, while the outcome matrix names the cells themselves. The empirical table is a sample from the joint distribution over these cells.

The smallest non-trivial universe

The input domain $\{\top, \bot\}$ has exactly two elements. We call this set the *Boolean universe* and write it $U = \{\top, \bot\}$, borrowing *universe* from set-theoretic usage as the ambient set over which subsets are formed and predicates evaluated. It is the smallest universe that supports a non-trivial predicate: one element admits no discrimination, two already admit the question whether an input is one value or the other. Part I develops the algebra of this universe and nothing larger. Every richer universe (multi- bit inputs, finite enumerations, arbitrary algebraic types) is assembled from Booleans by

the Kronecker products of Part II and beyond, so the algebra of these four cells governs systems that look more complicated but reduce, on inspection, to compositions of noisy bits.

The deferred joint distribution

A fully formal description assigns the four cells a joint distribution: given the rates of Section 1.2 and the prevalence of $\top$ in the input stream, the cells receive four probabilities summing to one. We defer that step. The formal object whose instances are exactly the noisy bits with two free rate parameters is defined in Section 4.5, which earns the joint distribution by first stating the two independence axioms that make the type well-posed and then proving the Kronecker factorization that justifies the parameter economy. Writing the joint mass here would either duplicate that machinery or assert it without support. Definition 4.5.1 gives the joint table, the channel matrix, and the worked biased-coin instance.

For one concrete instance, take the biased coin with $\varepsilon = 0.1$ and $\omega = 0.2$. Of its $\top$-trials a fraction $\tau = 0.8$ land in the correct-positive cell and $\omega = 0.2$ in the false-negative cell; of its $\bot$-trials a fraction $\eta = 0.9$ land in the correct-negative cell and $\varepsilon = 0.1$ in the false-positive cell. These are conditional shares within each row; converting them to outright cell probabilities requires the prevalence of $\top$, which is exactly the joint distribution deferred to Section 4.5. The next section estimates the rates from data without waiting for that formalization.

1.4 Sampling and Estimating Rates

In practice the rates are not given: a classifier arrives without a label stating its false-positive rate, and a channel must be characterized from observed traffic. We state how to estimate ε and ω from a finite sample and how to quantify the uncertainty. The full theory of these estimators (bias, variance, efficiency, and interval forms) is developed in Chapter 7.

Binomial estimators

Run a noisy bit n times, with n_0 trials of truth $\bot$ and $n_1 = n - n_0$ of truth $\top$. Each $\bot$-trial reports $\top$ independently with probability ε, so the false-positive count is binomial, $\mathrm{FP} \sim \mathrm{Binomial}(n_0, \varepsilon)$, and likewise $\mathrm{FN} \sim$

Binomial(n_1, ω). The row-conditional proportions

$$\hat{\varepsilon} = \frac{\mathrm{FP}}{n_0}, \qquad \hat{\omega} = \frac{\mathrm{FN}}{n_1}$$

are unbiased, $\mathbb{E}[\hat{\varepsilon}] = \varepsilon$, with variances

$$\mathrm{Var}(\hat{\varepsilon}) = \frac{\varepsilon(1-\varepsilon)}{n_0}, \qquad \mathrm{Var}(\hat{\omega}) = \frac{\omega(1-\omega)}{n_1}.$$

The variance shrinks as $1/n_0$, and its numerator $\varepsilon(1-\varepsilon)$ peaks at $\varepsilon = 1/2$, where the noisy bit is hardest to characterize, and vanishes at $\varepsilon \in \{0, 1\}$, where one trial settles the matter.

A confidence interval

For large n_0 the central limit theorem makes $\hat{\varepsilon}$ approximately Normal with mean ε and variance $\varepsilon(1-\varepsilon)/n_0$. Substituting the estimate into the variance gives the Wald interval at level $1 - \alpha$,

$$\hat{\varepsilon} \;\pm\; z_{\alpha/2}\sqrt{\frac{\hat{\varepsilon}(1-\hat{\varepsilon})}{n_0}},$$

with $z_{\alpha/2}$ the upper-$\alpha/2$ standard-Normal quantile. The Wald form is asymptotically correct but under-covers at small n_0 when ε is near 0 or 1, and can even leave $[0, 1]$; the Wilson, Clopper-Pearson, and Jeffreys intervals handle the boundary better, and Chapter 7 develops them.

Reproducibility

The companion notebook `bernoulli-py/notebooks/ch01-noisy-bit.ipynb` reproduces these claims: it estimates ε from a simulated binary symmetric channel across sample sizes spanning several orders of magnitude, exhibiting the $1/\sqrt{n_0}$ concentration of $\hat{\varepsilon}$ about the true rate; and it measures the empirical coverage of the 95% Wald interval, showing it approach nominal at moderate n_0 and fall below nominal as n_0 shrinks or ε approaches a boundary, the under-coverage noted above.

Remark (Independence is separate from rate-stability)**.** *The binomial model assumes the per-query error events are independent. Independence is a distinct assumption from the stability of the per-input rates: a system can have stable rates ε and ω yet errors that are correlated across queries. The point*

estimates $\hat{\varepsilon}, \hat{\omega}$ remain valid in that case, but the variances above, and every confidence interval and sample-size requirement built on them, misstate the true uncertainty. Independence is named as Axiom 1 in Section 4.2, and chapter 16 takes correlated-error systems as an explicit subject. △

1.5 Why "Atom"?

The noisy bit is atomic in two senses. First, it has no internal Bernoulli structure to decompose: the input is one Boolean value, the output is one Boolean value, and the entire behavior is two numbers, ε and ω. A smaller part would require a sub-Boolean input or output, and there is nothing below a Boolean. The atom is where internal accounting stops and external accounting begins, and every larger object in the book, chained channels, noisy logic gates, randomized algorithms, approximate sets, general algebraic types, is assembled from noisy bits by a specific composition rule. Second, the noisy bit sits at the minimum of the model-order ladder that Section 2.4 defines, the number of distinct error regimes an input domain splits into. A general noisy bit has model order two, the blocks $\{\top\}$ and $\{\bot\}$ carrying independent rates; the symmetric case collapses to order one. No noisy system over a Boolean domain has a smaller non-trivial channel matrix.

The claim is a working hypothesis until discharged, and chapters 2 through 4 do so in three directions. Chapter 2 gives the noisy bit a 2×2 row-stochastic matrix Q, places the symmetric, asymmetric, and Z-channel cases on the model-order ladder, and shows the atom closed under serial composition (the matrix product of row-stochastic matrices is row-stochastic). Chapter 3 asks what AND, OR, and NOT do to noisy bits and answers with the composition theorem (Section 3.3): the result is again a noisy bit, its rates closed-form in the operands'. One round of the Miller-Rabin primality test is a noisy bit with $\omega = 0$ and $\varepsilon \leq 1/4$, and the k-round bound $\varepsilon^{(k)} \leq (1/4)^k$ is the composition theorem applied k times rather than a separate analysis (Section 3.4). Chapter 4 names the two independence assumptions used silently until then, element-wise independence (Section 4.2) and conditional independence of per-block rates (Section 4.3); proves the Kronecker factorization they purchase (Section 4.4), which is what makes the per-element description sufficient; and writes the formal definition Bernoulli[bool] $= (X, Y, Q)$ (Section 4.5), of which every concrete object in chapters 1 through 3 is an instance.

Beyond Part I the atom is held fixed and its surroundings vary. Part II

keeps the Boolean codomain and arranges many noisy bits into a structure: a set S over a universe induces a membership query per element, an approximate set makes each query a noisy bit (the Bloom filter being the canonical one-sided instance, $\omega = 0$ with tunable ε), and the development is the algebra of how set operations propagate error, how classification measures quantify quality, and how much space a target accuracy demands. Part III replaces the Boolean codomain with an arbitrary algebraic type, lifting the atom to approximate maps, relations, and structured values, where the two-element model order ceases to be a ceiling. The atom is the unit that gets composed; everything that follows is composition.

Bibliographic Notes

Shannon and the channel framing. The noisy-channel reading of Face 2 in Section 1.1, and the binary symmetric channel that opens Chapter 2 (Section 2.1), descend directly from Shannon [Sha48]. Shannon's 1948 paper introduced the discrete memoryless channel as the mathematical object for noisy communication: a conditional distribution $P(\text{output} \mid \text{input})$ together with a capacity defined as the supremum of the mutual information between input and output. The binary symmetric channel is among the simplest of Shannon's channels, and it remains the standard pedagogical entry point to the channel framework, both in his original paper and in essentially every information-theory textbook that has followed.

The contingency-table tradition. The 2×2 contingency table that anchors Section 1.2 predates the modern probability framing of this book by decades. It grew up in epidemiology, biostatistics, and medical diagnostic testing, where the natural objects of study are tests with a true-positive (sensitivity) and a true-negative (specificity) rate, and where the population prevalence of the underlying condition is part of every working calculation. The vocabulary of sensitivity, specificity, and the cells of the 2×2 table travelled from clinical screening into machine learning largely unchanged; what differs across fields is which marginal the practitioner conditions on, and which derived rate gets the spotlight. The cross-walk at the end of Section 1.2, from (ε, ω) to precision and recall, is the modern statement of a much older bilingual conversation between epidemiology and signal detection.

Precision, recall, and information retrieval. The vocabulary of *precision* and *recall* arrived in this book by a different route, from information retrieval. In the IR setting one issues a query against a corpus and asks: of the documents returned, what fraction are relevant (precision); of the relevant documents in the corpus, what fraction were returned (recall). The textbook reference is Manning, Raghavan, and Schütze [MRS08], whose Chapter 8 develops the precision-recall vocabulary, the precision-recall curve, and the F-measure as the standard combined score. Chapter 1's emphasis on the difference between (ε, ω) and (precision, recall), that the first pair is intrinsic to a noisy bit and the second depends also on the workload, is the contingency-table analogue of the IR distinction between a system's intrinsic discriminating ability and its observed behaviour on a given query mix.

Miller, Rabin, and the primality tease. The Miller-Rabin algorithm previewed in Section 1.5 and developed in Section 3.4 has two distinct intellectual parents. Miller [Mil76] introduced the witness-based structure under the Extended Riemann Hypothesis; Rabin [Rab80] removed the hypothesis by randomizing the choice of witness and proved the unconditional $1/4$ bound on the per-round false-positive rate. Chapter 3's reading of Miller-Rabin, one round of the test as a Bernoulli[bool] with $\omega = 0$ and $\varepsilon \leq 1/4$, the k-round protocol as a composition, restates that result in the language of this text. The full historical record, the proof of the $1/4$ bound, and the companion treatment of randomized algorithms more broadly are taken up in Section 3.5.

The Bernoulli papers. The atom of this chapter, the noisy bit, is the order-2 Boolean instance of the formal Bernoulli framework developed in our companion papers. The two-axiom model and the rate-propagation theorems originate in Towell [Tow26b]; the extension from approximate sets to approximate values over arbitrary algebraic types, and, in particular, the treatment of Bernoulli[bool] as a value-with-error type, is the subject of Towell [Tow26j]. The present monograph [Tow26c] draws that material into a single development, starting from the noisy bit and building outward to sets, maps, relations, and the algebraic types the companion paper studies in full generality. Chapter 1's contribution is the decision to treat the biased coin, the binary symmetric channel, and the classifier verdict as three faces of one object and to call that object an atom; almost every other ingredient of the chapter has a longer history in one of the traditions noted above.

Chapter 2

Binary Channels and Model Order

2.1 The Binary Symmetric Channel

The three framings of Section 1.1 share a common structure, and the binary symmetric channel is that structure stripped to its essentials: a single Boolean input, a single Boolean output, and one crossover probability that the output disagrees with the input.

Definition 2.1.1 (Binary Symmetric Channel)**.** *A* binary symmetric channel *(BSC) with crossover probability* $\varepsilon \in (0,1)$ *is a random map* $\phi : \{\top, \bot\} \to \{\top, \bot\}$ *with*

$$\Pr\{\phi(x) \neq x\} \;=\; \varepsilon \quad \textit{for every } x \in \{\top, \bot\}.$$

The defining property is that the crossover is the same for both inputs: $\Pr\{\phi(\top) \neq \top\} = \Pr\{\phi(\bot) \neq \bot\} = \varepsilon$, so a single parameter plays the role of both the false-positive and the false-negative rate, $\varepsilon = \omega$. The symmetry is a genuine constraint, and most real systems lack it: a spam filter flags legitimate mail far more often than it leaks spam. The BSC is the clean starting point precisely because it isolates bit-flip noise from any question of which direction is noisier; the asymmetric generalization is Section 2.2.

The channel matrix

We record a binary channel as a 2×2 row-stochastic matrix Q, the *channel matrix*. The convention, fixed for the rest of the book: rows are indexed by the input block, row 1 for $B_1 = \{\top\}$ (positives) and row 2 for $B_2 = \{\bot\}$

(negatives); columns by the output label, $\top$ then $\bot$; and the (i, j) entry is the probability of output j given an input in block B_i. The four entries name the confusion-table rates, so the general order-2 channel and the BSC special case are

$$Q \;=\; \begin{pmatrix} \tau & \omega \\ \varepsilon & \eta \end{pmatrix} \;=\; \begin{pmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{pmatrix}, \qquad Q_{\mathrm{BSC}} \;=\; \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{pmatrix}.$$

The BSC matrix is *circulant*: its two rows are permutations of each other, the matrix expression of symmetry. At $\varepsilon = 0.05$, for instance, a positive input reads back correctly with probability 0.95 and flips with probability 0.05, and likewise for a negative input; under a uniform input prior the marginal error rate is exactly ε, independent of the prior because the channel is symmetric.

Capacity

The BSC's *capacity*, the maximum mutual information $I(X;Y)$ over input priors, is

$$C \;=\; 1 - H_2(\varepsilon), \qquad H_2(\varepsilon) = -\varepsilon \log_2 \varepsilon - (1-\varepsilon)\log_2(1-\varepsilon),$$

attained at the uniform prior $p = 1/2$. The derivation is short: $H(Y \mid X) = H_2(\varepsilon)$ for every prior, while $H(Y)$ is maximized at $p = 1/2$, so $C = \max_p H(Y) - H_2(\varepsilon) = 1 - H_2(\varepsilon)$. Capacity is one bit per use at $\varepsilon = 0$ (a perfect channel) and zero at $\varepsilon = 1/2$, where the output is independent of the input. The $\varepsilon = 1/2$ point is the zero-information case met in Section 1.2; for the BSC it coincides with the symmetric crossover, whereas for an asymmetric channel the zero-information locus is $\varepsilon + \omega = 1$.

The three faces in matrix form

The framings of Section 1.1 are the same matrix Q_{BSC} with different labels. The biased coin is a coin with one flip probability ε for both faces; the noisy line is Shannon's original BSC [Sha48], a bit flipped independently with probability ε; the classifier with equal false-positive and false-negative rates is a BSC, though real classifiers rarely satisfy $\varepsilon = \omega$ and are better modeled by the asymmetric channel of Section 2.2. Composing a BSC with itself makes matters worse, not better: two crossover-ε channels in series give crossover $2\varepsilon(1-\varepsilon) > \varepsilon$ for $0 < \varepsilon < 1/2$, derived in Section 2.3.

Model order of the BSC

The source literature uses two phrases for the BSC, "first-order" and "the symmetric special case of the order-2 model," and they refer to different things. Model order, formalized in Section 2.4, counts the blocks in the partition of the input domain that groups inputs sharing a common error rate. For the BSC both inputs share the crossover ε, so they form a single block and the BSC has *model order 1*; the asymmetric channel, with $\varepsilon \neq \omega$, keeps them in separate blocks at model order 2. Calling the BSC "the symmetric special case of the order-2 model" refers not to its order but to the family it is embedded in: one may always write the BSC with the two-block discrete partition and impose $\varepsilon = \omega$, the working frame for chapters 2 through 4. The BSC is genuinely model order 1, studied as a constrained member of the order-2 family; Section 2.4 separates model order from parameter count and proves that no binary channel exceeds order 2.

Throughout, ε is the false-positive rate, ω the false-negative rate, with $\tau = 1-\omega$ and $\eta = 1-\varepsilon$ the complementary rates, and for the BSC the single crossover $\varepsilon = \omega$. The source papers sometimes pair (FPR, TPR) rather than (FPR, FNR); since TPR $= 1 -$ FNR the choice is cosmetic, and this book uses (ε, ω).

2.2 The Asymmetric Binary Channel

The BSC's equal error rates are rarely satisfied in practice: a spam filter, a Bloom filter, and a medical screening test all treat their two error directions differently. Allowing the false-positive rate ε and the false-negative rate ω to vary independently gives the general binary channel.

Definition 2.2.1 (General asymmetric binary channel)**.** *An* asymmetric binary channel *with rates* $\varepsilon, \omega \in [0, 1)$ *is a random map* $\phi : \{\top, \bot\} \to \{\top, \bot\}$ *with* $\Pr\{\phi(\top) \neq \top\} = \omega$ *and* $\Pr\{\phi(\bot) \neq \bot\} = \varepsilon$*, and channel matrix*

$$Q \;=\; \begin{pmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{pmatrix}$$

in the row convention of Section 2.1.

Setting $\omega = \varepsilon$ recovers the BSC of Definition 2.1.1. Setting $\omega = 0$ makes row 1 equal to $(1, 0)$, so a positive input is always reported correctly: this is the Z-channel.

Definition 2.2.2 (Z-channel)**.** *A* Z-channel *with false-positive rate* $\varepsilon \in (0,1)$ *is an asymmetric binary channel with* $\omega = 0$*, channel matrix*

$$Q_Z = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1-\varepsilon \end{pmatrix}.$$

The Z-channel makes false negatives impossible while admitting false positives at rate ε, so an output of $\perp$ is fully reliable and an output of $\top$ is uncertain.[1] This reliable-no, uncertain-yes profile is exactly the Bloom filter: it answers set membership with no false negatives ($x \in S$ always returns $\top$) and a tunable false-positive rate ($x \notin S$ may still return $\top$), so a Bloom filter is a Z-channel, $\omega = 0$ with ε set by its bit-width and hash count. The doubly-biased coin, the screening test (small ω, larger ε) and its confirmatory counterpart (the reverse), and any system tuned by a decision threshold are all asymmetric channels; as the threshold sweeps, (ε, ω) traces a curve in $[0,1]^2$, each point a distinct channel, which is the ROC curve re-axed.

For one instance, take $\varepsilon = 0.05$, $\omega = 0.10$:

$$Q = \begin{pmatrix} 0.90 & 0.10 \\ 0.05 & 0.95 \end{pmatrix},$$

with false negatives twice as likely as false positives. Under a uniform input prior its accuracy is the diagonal average $\frac{1}{2}(0.90 + 0.95) = 0.925$. A BSC at $\varepsilon = \omega = 0.075$ achieves the same accuracy with both off-diagonal entries equal, so accuracy alone does not see the asymmetry; the distinction matters once the prior is non-uniform or the two errors carry different costs, which is the subject of Chapter 7.

The general order-2 case

The asymmetric channel is the general model-order-2 binary channel: the input domain partitions into $B_1 = \{\top\}$ and $B_2 = \{\perp\}$, each block carrying its own rate, ω for B_1 and ε for B_2. The BSC imposes $\omega = \varepsilon$, which merges the two blocks into one and drops the model order to 1 (Section 2.4); the Z-channel keeps the blocks distinct ($\omega = 0 \neq \varepsilon$) and so remains order 2 with a single free parameter, the first sign that model order and parameter count are separate quantities. Section 2.4 makes the distinction precise and

[1] Information-theory texts often take the opposite asymmetry ($\omega > 0$, $\varepsilon = 0$); the two are equivalent up to relabeling. The convention here aligns with positive approximate sets, where the false-positive direction is the natural error mode.

proves the Boolean ceiling: with only two input elements, no binary channel exceeds model order 2.

Remark (The BSC is a measure-zero special case)**.** *In the parameter square* $[0,1)^2$ *of pairs* (ε,ω)*, the binary symmetric channels are exactly the diagonal* $\varepsilon=\omega$*, a one-dimensional, measure-zero subset. A generic asymmetric channel is not a BSC, and an arbitrarily small perturbation off the diagonal turns a BSC into a two-block, model-order-2 channel. Symmetry is the exception in the space of binary channels, not the rule, which is why the book works in the order-2 frame and treats the BSC as the constrained case rather than the reverse.* △

2.3 Composition of Channels

Many systems chain noisy steps: an amplifier feeds another, a filter feeds a ranker. If an element passes through Q_1 and its output enters Q_2, the end-to-end channel is the matrix product.

Definition 2.3.1 (Serial composition of binary channels)**.** *For binary channel matrices* Q_1 *and* Q_2*, the* serial composition *applying* Q_1 *then* Q_2 *has channel matrix* $Q_{\text{total}}=Q_2Q_1$*, with* $(Q_{\text{total}})_{ij}=\sum_k (Q_1)_{ik}(Q_2)_{kj}$*.*

The sum is the law of total probability over the intermediate value k: $(Q_1)_{ik}$ carries block B_i to intermediate output k, and $(Q_2)_{kj}$ carries k to final output j. The right-to-left order Q_2Q_1 is the usual matrix convention, rightmost factor first.

Composing two BSCs

Two BSCs in series produce a third. Multiplying circulant matrices preserves the circulant form, and the off-diagonal entry of Q_2Q_1 is

$$\varepsilon_{\text{total}} \;=\; \varepsilon_1(1-\varepsilon_2)+(1-\varepsilon_1)\varepsilon_2 \;=\; \varepsilon_1+\varepsilon_2-2\varepsilon_1\varepsilon_2. \tag{2.1}$$

The composed error is the exclusive-or of the two individual flips: the input is corrupted exactly when one channel flips and the other does not (if both flip, the errors cancel). For $0<\varepsilon_1,\varepsilon_2<1/2$ this exceeds both operands, so two imperfect symmetric channels in series are noisier than either alone; at $\varepsilon_1=0.05$, $\varepsilon_2=0.10$ the composed crossover is 0.14. Chaining copies of one BSC drives the crossover toward the stationary $1/2$, where the channel carries no information.

Composing asymmetric channels

The general rule is the same matrix product. For Q_1, Q_2 with rates $(\varepsilon_i, \omega_i)$, the $(1,2)$ and $(2,1)$ entries of Q_2Q_1 give

$$\omega_{\text{total}} = (1-\omega_2)\omega_1 + \omega_2\varepsilon_1, \qquad \varepsilon_{\text{total}} = (1-\varepsilon_2)\varepsilon_1 + \varepsilon_2(1-\omega_1),$$

the channel-matrix form of the rate-propagation theorem of Towell [Tow26b]. Composition of asymmetric channels is not generally symmetric: the BSC is the special case in which symmetry is preserved because both factors are symmetric, forcing $\varepsilon_{\text{total}} = \omega_{\text{total}}$.

Serial versus parallel

Serial composition must not be confused with the parallel query structure of Section 4.4. Serial composition feeds one element's output into the next channel and yields a 2×2 matrix, the ordinary product Q_2Q_1. Parallel querying applies m independent channels to m distinct elements and yields a $2^m \times 2^m$ joint matrix, the Kronecker product $Q_1 \otimes \cdots \otimes Q_m$, which Theorem 4.4.1 shows is forced by the independence axioms. The two answer different questions, end-to-end error of a pipeline versus joint distribution of simultaneous queries, and combine when k multi-stage pipelines run in parallel: the Kronecker product across pipelines, the ordinary product within each. The serial product needs only the law of total probability; the Kronecker product needs the axioms (Corollary 4.4.1.1).

Algebraic properties

Serial composition is associative, since matrix multiplication is, so a chain of channels groups unambiguously. It is not commutative in general: $Q_2Q_1 \neq Q_1Q_2$ unless the factors share a symmetry. Two BSCs do commute, because (2.1) is symmetric in ε_1 and ε_2; a BSC and an asymmetric channel generally do not, as a direct computation of Q_2Q_1 and Q_1Q_2 shows, so the order of stages in a mixed pipeline matters.

Section 3.3 reframes the serial result algebraically: the hit rate of k composed one-sided stages satisfies $\eta_{\text{total}} = 1 - \prod_{i=1}^{k}(1-\eta_i)$, the Boolean-algebra restatement of the matrix product computed here.

2.4 Model Order

The BSC carries one crossover parameter and the asymmetric channel two, one per input. Whether a binary channel could carry more is a question

about the input domain, answered by *model order*.

Partitions and blocks

A *partition* of a set S is a collection of non-empty, pairwise disjoint blocks $B_1, \ldots, B_n$ whose union is S. In the channel model the relevant partition is of the input domain: elements of one block share a per-block error rate, and Axiom 2 (Section 4.3) treats the per-block rates $\{\alpha_1, \ldots, \alpha_n\}$ as the channel's free parameters.

Definition 2.4.1 (Model order)**.** *Let $\phi : D \to D$ be a binary channel on input domain D. A partition $\{B_1, \ldots, B_n\}$ of D is* compatible *with ϕ when every element of a block shares one error probability,*

$$\Pr\{\phi(x) \neq x\} = \alpha_i \quad \textit{for all } x \in B_i.$$

The model order *of ϕ is the number of blocks in the* coarsest *compatible partition: distinct error probabilities are placed in distinct blocks, and elements are grouped only when they genuinely share a rate. Model order is a property of the channel, the number of distinct error regimes its input domain carries, independent of any particular parameterization.*

The Boolean ceiling

The Boolean domain $D = \{\top, \bot\}$ admits exactly two partitions: the *trivial* one, $\{\{\top, \bot\}\}$, with both inputs in a single block sharing one rate; and the *discrete* one, $\{\{\top\}, \{\bot\}\}$, with each input carrying its own rate.

Theorem 2.4.1 (Boolean ceiling)**.** *Any binary channel with input domain $\{\top, \bot\}$ has model order at most* 2*.*

Proof. A partition of a set of cardinality k has at most k blocks, since each block is non-empty and the blocks are disjoint. For $D = \{\top, \bot\}$ the only partitions are the trivial (one block) and the discrete (two blocks), so the coarsest compatible partition has one or two blocks. □

The BSC realizes order 1: both inputs share the crossover ε, so the trivial partition is compatible and is the coarsest such, with channel matrix Q_{BSC} from Definition 2.1.1. The asymmetric channel of Definition 2.2.1 realizes order 2: with $\varepsilon \neq \omega$ the two inputs cannot share a block. The ceiling is tight (order 2 is attained) but not forced (the BSC sits at order 1). Serial composition stays within it: composing binary channels (Section 2.3) yields a binary channel, still of model order at most 2.

Model order versus parameter count

Model order is not the same as the number of free parameters, and conflating them is a common error. Model order counts partition blocks; parameter count counts the real numbers needed to specify the channel. For a single Boolean query an n-th order channel has n free parameters, one per block, so the two agree, but they part company in two ways. First, a constraint can reduce the parameter count below the order: the Z-channel of Definition 2.2.2 has model order 2 (its blocks carry the distinct rates 0 and ε) yet only one free parameter, since $\omega = 0$ is fixed. Second, in the multi-query setting the parameter count grows with the number of queries while the model order does not: a general order-n channel on m joint inputs carries $n(2^m - 1)$ parameters in the per-source-block convention, which the independence axioms collapse to nm by Kronecker factorization (Corollary 4.4.1.2), all while the model order stays n.

Channel	α_1 ($B_1 = \{\top\}$)	α_2 ($B_2 = \{\bot\}$)	order
BSC	ε	ε ($= \alpha_1$)	1
Z-channel	0	$\varepsilon \neq 0$	2
Asymmetric	ω	ε (free)	2
Perfect channel	0	0	1

The BSC and the perfect channel merge their two blocks (equal rates) and sit at order 1; the Z-channel and the asymmetric channel keep distinct rates and sit at order 2. The Z-channel row is the clean separation of the two counts: order 2, one parameter.

Axiom 2 and the n-ary ceiling

The partition-block structure is exactly what Axiom 2 (Axiom 2) acts on: it asserts that the per-block rates are conditionally independent given the objective set, and the model order is the dimension of that independence claim. In the order-2 Boolean case this is the independence of ε and ω. The Boolean ceiling bounds the claim at two factors, which is what keeps the framework tractable: every Boolean channel is at most two parameters per query, and the Kronecker factorization reduces the m-query count to $2m$. The machinery is not special to Booleans. For an input domain of cardinality k the same partition argument gives a ceiling of k, the subject of Chapter 9; Part I develops the order-2 Boolean case, from which the general theory is the natural lift.

Bibliographic Notes

Shannon and the channel framework. The channel matrix formalism used throughout this chapter descends directly from Shannon [Sha48]. Shannon's paper introduced the discrete memoryless channel as the mathematical object for studying noisy communication: a channel is specified by a conditional distribution $P(\text{output} \mid \text{input})$, and the capacity of the channel is the maximum mutual information between input and output over all input distributions. The binary symmetric channel of Section 2.1 is one of Shannon's central running examples; the crossover parameter ε and the capacity formula $1 - H(\varepsilon)$ are both in that 1948 paper. The treatment of the asymmetric binary channel in Section 2.2 is a natural extension of Shannon's framework to the case where the two input symbols face independent error probabilities. For a modern textbook development of binary channels, channel capacity, and the channel coding theorem, see Cover and Thomas [CT06], especially Chapters 7 and 8.

The Bernoulli papers. The connection between binary channels and the Bernoulli framework is drawn in Towell [Tow26b], which introduces the two-axiom model and proves the composition theorem that Section 2.3 reformulates in matrix-product language. The matrix-product form, that serial composition of two binary channels corresponds to matrix multiplication of their channel matrices, is a reorganization of the rate-propagation formula in that paper; see the companion notes in Section 4.6 for a precise account of what is new in this text and what descends from the companion papers.

The model-order treatment of Section 2.4, and in particular the definition of model order as the number of partition blocks in the coarsest compatible partition of the input domain (Definition 2.4.1), follows Towell [Tow26j]. That paper extends the Bernoulli framework from Boolean sets to approximate values over arbitrary algebraic types; the order-2 Boolean case developed in this chapter is the simplest instance of its general theory. The Boolean Ceiling Theorem (Theorem 2.4.1), that any binary channel over $\{\top, \bot\}$ has model order at most 2, is explicit in the algebraic-types paper as the base case of the more general statement that a channel over a k-element domain has model order at most k.

Model order versus degrees of freedom. A clarification that deserves explicit attention is the distinction between model order and parameter count, developed in Section 2.4 under the heading "Model order vs. parameter count." This distinction is easy to miss in the source papers, because

for a single Boolean query the two quantities agree: an n-th order channel on one Boolean input has n partition blocks and also n free parameters, one per block. The difference surfaces in the multi-query setting. When m queries are issued jointly, a general order-n channel on m inputs has $n(2^m - 1)$ parameters in the per-source-block convention, but the independence axioms (Axioms 1 and 2) force a Kronecker factorization that reduces this to nm. The model order does not change: it remains n, determined by the partition of the single-query input domain. What changes is the parameter count, and the axioms are entirely responsible for the reduction. Model order constrains which rows of the channel matrix Q can be identified; the axioms constrain how the channel matrix factorizes across queries. These are orthogonal constraints.

Further reading. Chapter 4's Bibliographic Notes (Section 4.6) provide additional context for the Kronecker product and its role in the multi-query factorization, and for the relationship between the Bernoulli axioms and the memoryless channel. Readers who want a deeper grounding in information theory beyond what Part I requires will find Cover and Thomas [CT06] the standard reference.

Chapter 3

Boolean Algebra Under Noise

3.1 AND Under Noise

A Bernoulli[bool] instance is a noisy bit with rates (ε, ω) (Definition 4.5.1). The first question of the chapter is what a Boolean operation does to two of them. AND is the natural start: it is the operation behind a Bloom filter's k-hash query and Miller-Rabin's k-round test, and OR and NOT follow from it (Section 3.2).

Let $\tilde{b}_1, \tilde{b}_2$ be independent instances with rates $(\varepsilon_i, \omega_i)$, and apply AND to their observed outputs; the true value is the AND of the true inputs. Two independence assumptions are in play, and the chapter's purpose is to invoke them deliberately. Axiom 1 (Axiom 1) makes the error events of the two instances independent, licensing every product below; Axiom 2 (Axiom 2) makes each instance's rates stable constants rather than a drifting process. Chapter 4 proves both and shows they are the content of the Kronecker factorization (Section 4.2, Section 4.3).

The AND of the observed outputs is $\top$ exactly when both report $\top$, which by Axiom 1 happens with the following probabilities across the four true inputs:

b_1	b_2	correct AND	Pr[observed AND $= \top$]
$\top$	$\top$	$\top$	$(1-\omega_1)(1-\omega_2)$
$\top$	$\bot$	$\bot$	$(1-\omega_1)\varepsilon_2$
$\bot$	$\top$	$\bot$	$\varepsilon_1(1-\omega_2)$
$\bot$	$\bot$	$\bot$	$\varepsilon_1\varepsilon_2$

Output rates

The true AND is $\top$ only in the first row, so a false negative (true $\top$, observed $\bot$) is the complement of that row's success:

$$\omega_\wedge \;=\; 1-(1-\omega_1)(1-\omega_2) \;=\; \omega_1+\omega_2-\omega_1\omega_2. \tag{3.1}$$

This depends only on the input rates, not on the input prior: the AND succeeds on a true $\top$ only if both instances do, and the probability that at least one misses is the inclusion-exclusion expression above.

The false-positive rate is different. The true AND is $\bot$ in the last three rows, and the observed AND is $\top$ there with conditional probabilities $(1-\omega_1)\varepsilon_2$, $\varepsilon_1(1-\omega_2)$, and $\varepsilon_1\varepsilon_2$. The effective false-positive rate is the prior-weighted mixture of these, conditioned on the true AND being $\bot$. Writing π for the prior that an input is $\top$,

$$\varepsilon_\wedge \;=\; \frac{\pi(1-\pi)\big[(1-\omega_1)\varepsilon_2+\varepsilon_1(1-\omega_2)\big]+(1-\pi)^2\varepsilon_1\varepsilon_2}{1-\pi^2}. \tag{3.2}$$

Unlike the false-negative rate, this depends on the input distribution. The two extremes are instructive. When the two inputs are jointly $\bot$ (the all-negative regime, $\pi \to 0$ or both inputs known negative), the boundary patterns vanish and

$$\varepsilon_\wedge \;=\; \varepsilon_1\varepsilon_2,$$

the product, which drives the false-positive rate down: two noisy bits both have to err for the conjunction to err. When one input is positive, the relevant pattern contributes $\approx \varepsilon$ (only one instance need err), so the mixture lies between $\varepsilon_1\varepsilon_2$ and $\max(\varepsilon_1,\varepsilon_2)$. The all-negative product regime is the one Miller-Rabin operates in: its k rounds all test the same composite, so the only true input is $\bot$ and the conjunction's false-positive rate is the clean product $\prod_i \varepsilon_i$ (Section 3.4).

AND therefore takes two Bernoulli[bool] instances to another, with a prior-free false-negative rate (3.1) and a false-positive rate that is the product in the all-negative regime and prior-dependent in general. For two observers of a fair coin with rates $(0.10, 0.05)$ and $(0.08, 0.12)$, requiring both to report heads gives $\omega_\wedge = 0.164$ by (3.1) and, at $\pi = 0.5$, an effective $\varepsilon_\wedge \approx 0.057$ by (3.2): AND trades false-negative accuracy for false-positive suppression.

3.2 OR and NOT

NOT is immediate and OR follows from it by De Morgan, completing the Boolean basis.

NOT

Complementing the output of a Bernoulli[bool] instance turns each false negative into a false positive and conversely: a $\bot$ reported on a true $\top$ becomes a $\top$ reported on the complemented true $\bot$. So NOT swaps the rate pair, $(\varepsilon, \omega) \mapsto (\omega, \varepsilon)$, which on the channel matrix is the row exchange

$$Q_{\neg} \;=\; \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \omega & 1-\omega \end{pmatrix}. \tag{3.3}$$

NOT is an involution on Bernoulli[bool]: applying it twice restores the rates, the noisy-bit form of $\neg\neg b = b$ (at the level of the channel matrix, not of any realized output).

OR

By De Morgan, $\tilde{b}_1 \vee \tilde{b}_2 = \neg(\neg\tilde{b}_1 \wedge \neg\tilde{b}_2)$. Negating the inputs swaps their rates, the AND of the negated instances has false-negative rate $\varepsilon_1+\varepsilon_2-\varepsilon_1\varepsilon_2$ by (3.1) with rates swapped, and the outer NOT swaps that into the OR's false-positive rate. The truth table confirms it directly: the observed OR is $\bot$ only when both instances report $\bot$, so a false positive (true $\bot\bot$, observed $\top$) is

$$\varepsilon_{\vee} \;=\; 1-(1-\varepsilon_1)(1-\varepsilon_2) \;=\; \varepsilon_1+\varepsilon_2-\varepsilon_1\varepsilon_2, \tag{3.4}$$

the inclusion-exclusion probability that at least one instance false-positives. This is prior-free. The OR's false-negative rate, dually, is prior-dependent and equal to the product $\omega_1\omega_2$ in the all-positive regime.

The duality is exact: AND has a prior-free false-negative rate (3.1) and a prior-dependent false-positive rate, while OR has a prior-free false-positive rate (3.4) and a prior-dependent false-negative rate. AND suppresses false positives and inflates false negatives; OR does the reverse; NOT exchanges the two regimes. In the symmetric case $\varepsilon_i = \omega_i = \varepsilon$, both inflate by the same $2\varepsilon - \varepsilon^2$, and De Morgan's duality is manifest.

Proposition 3.2.1 (Closure under Boolean operations)**.** *If* $\tilde{b}_1, \ldots, \tilde{b}_n$ *are* Bernoulli[bool] *instances, every Boolean formula built from them with AND,*

OR, and NOT is a Bernoulli[bool] *instance whose prior-free rate is a polynomial in the input rates and whose prior-dependent rate is a rational function of the input rates and the prevalences.*

Proof. Induct on the formula. NOT preserves Bernoulli[bool] by (3.3); AND does by (3.1) and (3.2); OR does by De Morgan from NOT and AND. Since $\{\wedge, \vee, \neg\}$ is a complete basis and every formula is a finite composition of these, the induction closes. □

Closure is what makes the framework composable: a Boolean circuit of noisy gates has, at every stage, a Bernoulli[bool] output whose error propagates in closed form, with the prior-dependence of the AND and OR rates inherited as in Section 3.1. The next section isolates the case where that propagation is cleanest, the chained one-sided gate, and states it as the composition theorem.

3.3 The Composition Theorem

The AND's false-negative rate (3.1) and the OR's false-positive rate (3.4) are both instances of one pattern: the probability that a chain produces no error factors as a product. This section states it as a theorem and isolates the regime where it is exact.

Theorem 3.3.1 (Composition theorem)**.** *Let $\hat{b}_1, \ldots, \hat{b}_k$ be the outputs of k chained identity-with-error stages, stage i correct with probability $1-\eta_i$, with error events independent across stages. If any single stage error corrupts the end-to-end output (the identity-with-error regime), then the end-to-end correctness probability and error rate are*

$$\eta_{\text{total}} = \prod_{i=1}^{k}(1-\eta_i), \qquad \eta_{\text{total}} = 1 - \prod_{i=1}^{k}(1-\eta_i). \tag{3.5}$$

Proof. By induction on k. The case $k = 1$ is immediate. For the step, the chain of k stages is correct iff the first $k-1$ are correct (probability $\prod_{i<k}(1-\eta_i)$, by hypothesis) and stage k is correct (probability $1-\eta_k$); by independence these multiply, giving $\prod_{i\leq k}(1-\eta_i)$. Complementation gives η_{total}. □

The inductive step uses "if and only if both," which holds only when any single error is fatal. If a later stage can accidentally undo an earlier error, the formula overcounts; this is the BSC caveat below.

Which rate each direction governs

The theorem applies to the error direction in which a single stage error propagates. For the conjunction of k instances (report $\top$ iff all do), a true $\top$ survives only if every stage reports $\top$, so correctness multiplies and the false-negative rate is

$$\omega_{\wedge,k} = 1 - \prod_{i=1}^{k}(1 - \omega_i),$$

the theorem with $\eta_i = \omega_i$. Dually, the disjunction's false-positive rate is $\varepsilon_{\vee,k} = 1 - \prod_i(1 - \varepsilon_i)$.

The opposite direction is the *dual*: the conjunction false-positives on an all-negative input only if *every* stage false-positives, so errors multiply and

$$\varepsilon_{\wedge,k} = \prod_{i=1}^{k} \varepsilon_i \tag{3.6}$$

in the all-negative regime. This is the mechanism behind Miller-Rabin: its k rounds all test the same composite, the only true input is $\bot$, and the false-positive rate is the product $\prod_i \varepsilon_i \leq (1/4)^k$ (Section 3.4). The two formulas are not interchangeable: the conjunction's false-negative rate is $1-\prod(1-\omega_i)$, but its false-positive rate is the product $\prod \varepsilon_i$, not $1 - \prod(1 - \varepsilon_i)$.

The BSC caveat

The theorem computes the probability that at least one stage errs. For the identity-with-error and one-sided cases this equals the end-to-end error rate, but the BSC violates the "single error is fatal" premise: two flips cancel. Two BSCs of crossover ε in series have end-to-end crossover $2\varepsilon - 2\varepsilon^2$ (the odd-flip probability, Equation (2.1)), whereas the theorem's $1 - (1 - \varepsilon)^2 = 2\varepsilon - \varepsilon^2$ counts the both-flip event as an error even though it returns the correct value. The two agree only when cancellation is impossible: one-sided channels ($\omega = 0$ or $\varepsilon = 0$), and the AND and OR gates in their respective monotone directions. For the BSC, the matrix product of Section 2.3 is the correct tool, not this theorem. The two formulas answer different questions about different models.

The Miller-Rabin regime, worked

Take k one-sided instances ($\omega_i = 0$), the Bloom-filter and Miller-Rabin setting, queried in the all-negative regime, with false-positive rates ε_i. The

conjunction has $\omega_{\wedge,k} = 0$ (no stage ever suppresses a true $\top$), and its false-positive rate is the product (3.6). For $\varepsilon = (0.20, 0.15, 0.10, 0.25)$,

$$\varepsilon_{\wedge,4} = 0.20 \cdot 0.15 \cdot 0.10 \cdot 0.25 = 7.5 \times 10^{-4},$$

far below any single stage: every stage must independently false-positive for the conjunction to. With all four at $\varepsilon = 1/4$ the product is $(1/4)^4 \approx 3.9 \times 10^{-3}$, the Miller-Rabin bound at $k = 4$. The conjunction of one-sided tests drives the false-positive rate down geometrically, which is exactly the mechanism Miller-Rabin exploits.

Remark (Independence is required)**.** *The product factorizations rest on independence across stages. If the stages are perfectly correlated, so that all err together or none do, the end-to-end error rate is the single-stage rate η regardless of k, not $1-(1-\eta)^k$ or η^k. Repeated queries of the same deterministic predicate on the same element are the extreme case: they are perfectly correlated, not independent, and neither the theorem nor its dual applies. Chain-stage independence is a structural property of the pipeline, supplied in the Miller-Rabin analysis by drawing distinct witnesses (Section 3.4); chapter 16 takes correlated stages as an explicit subject. It is distinct from Axiom 1 (Axiom 1), which governs independence across input elements rather than across pipeline stages, though in well-formed pipelines where each stage queries a distinct element the former follows from the latter.* $\triangle$

The parallel counterpart to this serial result is the Kronecker factorization (Theorem 4.4.1): correctness multiplies along a chain, while the joint confusion matrix of m parallel queries factors as $Q_1 \otimes \cdots \otimes Q_m$. The serial product needs only chain-stage independence; the Kronecker product needs Axiom 1.

3.4 Miller-Rabin

Miller-Rabin primality testing is a k-fold one-sided Bernoulli[bool] conjunction. Number theory contributes one fact, the per-round bound $\varepsilon \leq 1/4$; everything else is the composition machinery of §§3.1–3.3.

The witness test as a noisy bit

Fix an odd $n > 3$ and write $n-1 = 2^r d$ with d odd. A witness $a \in \{2, \ldots, n-2\}$ certifies compositeness if the sequence $a^d, a^{2d}, \ldots, a^{2^r d} \pmod{n}$ reaches 1 without passing through -1 and without starting at ± 1; otherwise the test

reports "probably prime." Identify "probably prime" with $\top$ and "composite" with $\bot$, and let the true value be whether n is prime. Then a prime always passes (by Fermat's little theorem and the fact that the only square roots of 1 modulo a prime are ± 1), so the test has no false negatives, $\omega = 0$. A composite passes only when a lies in its *strong-liar set*, and Rabin's theorem bounds that set at $(n-3)/4$, so a uniform witness gives

$$\varepsilon \;\leq\; \frac{1}{4}. \tag{3.7}$$

A single round is therefore a Bernoulli[bool] instance with $\omega = 0$ and $\varepsilon \leq 1/4$.

To bring the k-round test under Axiom 1 (Axiom 1), recast it as a predicate on the witness pool: $\tilde{P}(a)$ is the verdict for witness a, the pool $\{2, \ldots, n-2\}$ is the universe, and distinct witnesses are distinct input elements. The k-round test is the conjunction $\tilde{P}(a_1) \wedge \cdots \wedge \tilde{P}(a_k)$ over k independently drawn witnesses, exactly the operation of Section 3.1.

The k-round bound

Each round is one-sided with $\omega_i = 0$, so the conjunction has $\omega_k = 0$: the test never calls a prime composite. All k rounds test the same n, so the only true input is $\bot$ (for a composite), and the false-positive rate is the all-negative dual of the composition theorem, (3.6): the conjunction false-positives only if every witness is a strong liar, so

$$\varepsilon_k \;=\; \prod_{i=1}^{k} \varepsilon_i \;\leq\; \left(\frac{1}{4}\right)^{k}. \tag{3.8}$$

The equality is the product over the independently drawn witnesses (membership of a uniform draw in the strong-liar set is independent across draws); the inequality bounds each round by the Monier-Rabin value $\varepsilon_i \leq 1/4$. Both ingredients are needed: the size bound $|L_n| \leq (n-3)/4$ from number theory and the independent sampling that makes Axiom 1 apply. At $k = 5$ the bound is below 10^{-3}, and at $k = 20$ below 10^{-12}, effectively definitive.

A worked composite: $n = 341$

The number $341 = 11 \times 31$ is a Fermat pseudoprime to base 2 ($2^{340} \equiv 1 \pmod{341}$), so the Fermat test passes it, but Miller-Rabin does not. With $n - 1 = 2^2 \cdot 85$, take $a = 2$: $2^{85} \equiv 32 \pmod{341}$ with $32 \neq \pm 1$, and $32^2 \equiv 1 \pmod{341}$. A non-trivial square root of 1 has appeared, which a prime

Algorithm 1: Miller-Rabin primality test (k rounds)

input : n: odd integer > 3; k: number of rounds
output : COMPOSITE (definitive), or PROBABLY-PRIME (FPR $\leq (1/4)^k$)

```
function MillerRabin(n, k)
    write n − 1 = 2^r d with d odd;
    for i ← 1 to k do
        choose a uniformly from {2, ..., n − 2};
        x ← a^d mod n;
        if x = 1 or x = n − 1 then continue;
        for j ← 1 to r − 1 do
            x ← x^2 mod n;
            if x = n − 1 then break;
        end
        if x ≠ n − 1 then return COMPOSITE;
    end
    return PROBABLY-PRIME;
```

modulus forbids, so the witness $a = 2$ certifies 341 composite. Miller-Rabin is stronger than Fermat precisely because it inspects the square roots of 1 along the way to a^{n-1}, not just the endpoint.

The unification

Miller-Rabin is not a separate technique. The single round is a Bernoulli[bool] instance with $\omega = 0$ and $\varepsilon \leq 1/4$; the k-round test is the conjunction of Section 3.1; the bound $\varepsilon_k \leq (1/4)^k$ is (3.8); and the design rule, choose $k \geq \frac{1}{2}\log_2(1/\epsilon) = \log_4(1/\epsilon)$ to reach false-positive target ϵ, is that bound solved for k ($k = 5$ reaches 10^{-3} and $k = 20$ reaches 10^{-12}, the rounds quoted above). The number theory supplies the per-round $1/4$; the rest is Boolean algebra over noisy bits. The broader theory of randomized algorithms, including the Las Vegas versus Monte Carlo distinction, is Chapter 11; the companion one-sided example, the Bloom filter, is Section 3.5.

3.5 Composition is the Bridge to Sets

The Boolean algebra of Bernoulli[bool] is the foundation for the richer types of Parts II and III, and three concrete bridges carry it there.

Bloom filters are m-fold conjunctions. A Bloom filter reports membership by checking that all m hash positions are set, an m-fold AND of one-sided lookups. Each lookup has $\omega = 0$ (a member sets every position) and per-hash false-positive rate ε_i. In the all-negative regime of a non-member, the conjunction false-positives only if every hash collides, so by the dual (3.8) of the composition theorem the filter's false-positive rate is $\prod_{i=1}^{m} \varepsilon_i \leq p^m$, the classical Bloom formula, recovered as Boolean algebra over noisy bits. The structure is identical to Miller-Rabin; only the source of the per-stage rate differs, a design parameter here, a number-theoretic theorem there. Chapter 5 develops the filter and its parameter selection in full.

Set operations are Boolean operations on membership. For approximate sets $\tilde{A}, \tilde{B}$, membership $x \in \tilde{A}$ is a Bernoulli[bool] instance, and the set operations are pointwise Boolean operations on these: intersection is AND, union is OR, complement is NOT. The rate formulas of Section 3.1 and Section 3.2 therefore give, per element,

$$\omega_{\cap} = 1 - (1 - \omega_A)(1 - \omega_B), \quad \varepsilon_{\cup} = \varepsilon_A + \varepsilon_B - \varepsilon_A \varepsilon_B, \quad (\varepsilon_{A^c}, \omega_{A^c}) = (\omega_A, \varepsilon_A).$$

The intersection's false-positive rate is the product $\varepsilon_A \varepsilon_B$ for elements outside both sets (the bulk regime), and is boundary-dependent for elements in exactly one set, exactly the prior-dependence of the AND in Section 3.1. Chapter 6 carries the general case, including the rate-uniformity conditions an approximate set must satisfy for these per-element rates to define a single Bernoulli[S] instance, conditions that chapter 16 (Chapter 16) shows can fail.

The Boolean case is a fiber of the map algebra. Replacing the Boolean codomain with an arbitrary type Y lifts the same composition law: an approximate map $\tilde{f} : X \to Y$ has a channel matrix that grows from 2×2 to $|Y| \times |Y|$, but composition is still the matrix product and the independence structure is unchanged. The Boolean algebra of this chapter is the $Y = \text{bool}$ fiber of the map algebra that Chapter 9 develops. The composition law is the constant; only the codomain varies.

The bibliographic notes (Section 3.5) follow. Chapter 4 formalizes the independence axioms these derivations assumed, and Part II builds the full algebra of approximate sets and maps on both chapters.

Bibliographic Notes

Miller and Rabin. The primality test of Section 3.4 has two parents. Miller [Mil76] gave the witness structure, compute $a^d \bmod n$ and test for a non-trivial square root of unity, and a deterministic polynomial-time variant conditional on the Extended Riemann Hypothesis. Rabin [Rab80] removed the hypothesis by randomizing the witness and proving the unconditional bound: for any composite n, at most a quarter of the witnesses are strong liars. That $1/4$ is the one fact the framework borrows; the chapter's reading of a single round as a Bernoulli[bool] with $\omega = 0$ and $\varepsilon \leq 1/4$ restates Rabin's result in this book's language. Textbook treatments with the full number-theoretic argument are Cormen et al. [Cor+09, Ch. 31] and, for randomized algorithms broadly, Mitzenmacher and Upfal [MU17].

The Bernoulli papers. The composition theorem of Section 3.3 is the Boolean-algebra form of the rate-propagation result of Towell [Tow26b], restated for a single Bernoulli[bool] instance rather than a set, and the AND, OR, and NOT rules of Section 3.1 are the Boolean base case of the algebraic-type treatment in Towell [Tow26j].

One algebra, three lenses. A single composition algebra runs through chapters 2 through 4 at three abstraction layers: the *matrix product* $Q_2 Q_1$ for serial channels (Section 2.3); the *Boolean-algebra* form $\eta_{\text{total}} = 1 - \prod_i (1 - \eta_i)$ for chained one-sided gates (this chapter); and the *Kronecker product* $Q_1 \otimes \cdots \otimes Q_m$ for parallel queries (Theorem 4.4.1). The Boolean form differs from the matrix product for the BSC, where cancellation makes the two disagree, and coincides with it for one-sided channels. Chapters 2 and 4 situate the matrix-product and Kronecker layers in their own literatures (Section 2.4, Section 4.6); seeing all three as one algebra is a contribution of this text.

Chapter 4

Two Axioms and Kronecker Factorization

4.1 Why Axioms?

Chapters 1 through 3 derived the gate rates, the composition theorem (Section 3.3), and the Miller-Rabin bound, and at two points they used independence without naming it. This chapter names the two assumptions, states them precisely, and proves what their conjunction buys: the Kronecker factorization that collapses the parameter count of an m-query analysis from exponential to linear, and the formal definition of the type the first three chapters circled.

The first assumption is element-wise independence: error events at distinct inputs are independent. Every product of probabilities in chapters 1 through 3, the composition theorem's chain, Miller-Rabin's k independent witnesses (Section 3.4), the AND and OR truth tables of Section 3.1, rested on it. When it fails, as for two classifiers trained on shared data whose errors correlate, the product formulas become bounds at best. We call it Axiom 1 (Section 4.2).

The second is rate stability: each block's error rate is a fixed property, the same on the hundredth query as the first. The AND derivation treated ε and ω as constants, assuming the channel does not drift, warm up, or adapt. We call it Axiom 2 (Section 4.3). It rules out fatigue, history-dependence, and adversarial adaptation, and it is a real restriction, not a triviality. Together the two axioms force the joint confusion matrix of m queries to factor as a Kronecker product of m small per-query matrices (Section 4.4), after which the formal definition of Section 4.5 writes itself: a

triple (X, Y, Q) satisfying the two axioms, of which every example in chapters 1 through 3 is an instance.

4.2 Axiom 1: Element-Wise Independence

The first axiom governs the spatial structure of error: how the outcomes at distinct inputs relate. For an objective set $A \subseteq U$ and the approximate predicate $\tilde{A}$, write the error indicator $E_x = \mathbf{1}[\tilde{A}(x) \neq \mathbf{1}_A(x)]$ at each $x \in U$.

Axiom 1 (Element-wise independence)**.** *Given the objective set A and the error rates, the family $\{E_x : x \in U\}$ is mutually independent, and within each partition block the indicators are identically distributed.*

This is a property of the process generating the answers, not of any particular implementation, and its immediate consequence is that joint query distributions factor.

Proposition 4.2.1 (Factored joint distribution)**.** *Under Axiom 1, for any distinct $x_1, \ldots, x_n$ the joint distribution of the responses factors,*

$$\mathrm{P}\left(\tilde{A}(x_1) = s_1, \ldots, \tilde{A}(x_n) = s_n | A\right) \;=\; \prod_{i=1}^{n} \mathrm{P}\left(\tilde{A}(x_i) = s_i | A\right),$$

each factor being one of $\varepsilon, \eta, \tau, \omega$ according to x_i's block and the correctness of s_i.

Proof. Mutual independence of $\{E_x\}$ gives independence of any finite subfamily, so the joint over $\{x_1, \ldots, x_n\}$ factors; each $\tilde{A}(x_i)$ is a deterministic function of E_{x_i} and the exact answer $\mathbf{1}_A(x_i)$, which A fixes, so the factored form transfers to the responses. □

This proposition is the engine behind every multi-query formula in the book: a joint probability over distinct inputs written as a product of marginals is an application of it. It is also what the Bloom filter's analysis assumes (distinct non-members are independent false positives) and what Miller-Rabin's k-round bound needs (distinct witnesses are independent error events, Section 3.4). The Bloom filter is verified to satisfy Axiom 1 under the random-oracle model in Chapter 5.

Axiom 1 is a genuine modeling assumption, not an automatic consequence of each marginal being well-formed. Two classifiers trained on shared data are each a Bernoulli[bool], but their errors on a single negative input

correlate. If their error indicators have correlation ρ, the joint false-positive probability is

$$\mathrm{P}\left(E^{(1)} = 1, E^{(2)} = 1\right) = \varepsilon_1\varepsilon_2 + \rho\sqrt{\varepsilon_1(1-\varepsilon_1)\,\varepsilon_2(1-\varepsilon_2)},$$

which for $\varepsilon_1 = \varepsilon_2 = 0.10$ and $\rho = 0.60$ is 0.064, a factor of 6.4 above the product 0.01 that independence would predict. The marginals can be perfectly calibrated while the joint fails to factor; correlated errors are worse than independent ones in this precise, quantifiable sense. (The two-classifiers-on-one-input case is one step removed from Axiom 1, which governs distinct inputs to one predicate, but the arithmetic of the violation is identical.) The Kronecker theorem of Section 4.4 makes this exact at the level of the joint confusion matrix: with Axiom 1 the matrix has a structured product form, and without it the parameter count explodes.

4.3 Axiom 2: Conditional Independence of Block Error Rates

Axiom 1 governs error across distinct elements; Axiom 2 governs the random per-block rates. The model partitions the input domain into blocks $B_1, \ldots, B_n$ sharing a rate each (Section 2.4); for the Boolean domain the blocks are $B_1 = \{\top\}$, carrying ω, and $B_2 = \{\bot\}$, carrying ε.

Axiom 2 (Conditional independence of block error rates)**.** *The random per-block rates $\alpha_1, \ldots, \alpha_n$, conditional on the objective set A, are mutually independent: $f(\alpha_1, \ldots, \alpha_n \mid A) = \prod_i f(\alpha_i \mid A)$.*

Knowing the error rate on one block tells you nothing about another's. In the order-2 Boolean case this is the single statement that ε and ω are conditionally independent given A, the instance that arises throughout the book. The rates are random in the sense that they may vary across realizations of the data structure (a Bloom filter's false-positive rate depends on which elements were inserted); Axiom 2 constrains how those rates relate across blocks, not across elements within a block, which is Axiom 1's identically- distributed clause.

Proposition 4.3.1 (Stable channel matrices)**.** *Under Axiom 2 (with Axiom 1), the channel row for block B_i is the same at every query of an element of B_i, independent of the order or history of prior queries.*

Proof. The output distribution for an element of B_i depends only on the realized rate α_i, which Axiom 2 makes independent of the other blocks' rates and which is drawn once for the structure; by Axiom 1 the per-element error events are conditionally independent given α_i. The row of Q for B_i is therefore a fixed function of α_i at every query, with no dependence on prior outcomes, so it is stable across the sequence. □

This is the precise sense of "one channel matrix Q per block": the entry Q_{ij} does not drift or warm up, and a single Q describes an entire query sequence. A channel that violates this is a Markov-correlated one, where the rate after an error rises from a base ε_0 to an elevated ε_1. With $\varepsilon_0 = 0.05$ and $\varepsilon_1 = 0.30$, two consecutive negatives are both reported $\top$ with probability $0.05 \times 0.30 = 0.015$, six times the $\varepsilon_0^2 = 0.0025$ that a stable rate predicts. Error clustering of this kind is invisible to the factored model and is exactly what Axiom 2 forbids; a Markov channel must be handled by augmenting the state, after which the standard machinery applies to the augmented model.

Stability is what makes the Kronecker factorization possible (Section 4.4): the m-query joint matrix is a product of copies of one stable Q. The payoff is parametric. A general order-n channel on m joint queries needs $n(2^m - 1)$ parameters in the per-source-block convention; under the two axioms the Kronecker structure forces nm, with no cross-query terms. For the order-2 Boolean channel at $m = 50$ that is $2m = 100$ parameters rather than $2(2^{50} - 1) \approx 2.25 \times 10^{15}$. (One subtlety, taken up in Section 4.4: this count is exact for the rate-conditional matrix always, and for the unconditional matrix when rates are deterministic or no block is queried twice, Corollary 4.4.1.1; same-block queries on genuinely random rates incur a Jensen gap.)

4.4 The Kronecker Factorization Theorem

The two axioms together force a precise structure on the joint behavior of many queries. Index the m queries by position ℓ, with query ℓ on an element of block B_{j_ℓ} producing output k_ℓ. The *Kronecker product* $P \otimes Q$ replaces each entry P_{ij} by the scaled block $P_{ij}Q$; it is associative.

Theorem 4.4.1 (Kronecker factorization, conditional form)**.** *Let $\tilde{A}$ be an n-th order Bernoulli predicate satisfying Axiom 1 and Axiom 2. For any realization α of the per-block rates, let $Q_\ell^*(\alpha)$ be the per-query channel matrix at position ℓ. Then the rate-conditional joint confusion matrix of m*

queries on m distinct elements factors as

$$Q^{(m)*}(\alpha) \;=\; Q_1^*(\alpha) \otimes \cdots \otimes Q_m^*(\alpha).$$

Proof. Fix the rates α. By Proposition 4.3.1 the row of $Q_\ell^*(\alpha)$ for block B_{j_ℓ} is determined by α_{j_ℓ} alone, independent of other positions. Conditional on α, the outputs are deterministic functions of the per-element error events, which are mutually independent by Axiom 1, so the joint entry factors,

$$\mathrm{P}\left(\tilde{A}(x_1) = k_1, \ldots | \alpha\right) = \prod_{\ell=1}^{m} (Q_\ell^*(\alpha))_{j_\ell, k_\ell},$$

which is by definition the corresponding entry of $Q_1^*(\alpha) \otimes \cdots \otimes Q_m^*(\alpha)$. □

Corollary 4.4.1.1 (Unconditional factorization, distinct blocks)**.** *If the queried blocks $j_1, \ldots, j_m$ are pairwise distinct, the unconditional matrix factors as $Q^{(m)} = Q_1 \otimes \cdots \otimes Q_m$ with $Q_\ell = \mathrm{E}\left[Q_\ell^*(\alpha)\right]$.*

The conditional factorization holds always; the unconditional one needs the distinct-block hypothesis because taking the expectation through the product requires independent factors. When two queries hit the same block B_j, both involve the same random α_j, and $\mathrm{E}\left[\alpha_j^2\right] > (\mathrm{E}\left[\alpha\right]_j)^2$ opens a Jensen gap, so the unconditional joint exceeds the Kronecker prediction. Two regimes avoid this: deterministic rates (where $Q_\ell^* = Q_\ell$ and the conditional and unconditional forms coincide, the case of every worked example in this book) and pairwise-distinct queried blocks.

Corollary 4.4.1.2 (Parameter reduction)**.** *Under the two axioms, the rate-conditional m-query matrix needs nm parameters, against $n(2^m - 1)$ for an unconstrained m-query channel in the per-source-block convention; for the order-2 Boolean predicate, $2m$ against $2(2^m - 1)$. The same count holds for the unconditional matrix in the two regimes above.*

Proof. Each per-query matrix Q_ℓ^* is $n \times 2$ row-stochastic, contributing one free parameter per row, so n per query and nm across m queries with no shared parameters (Axiom 2). The unconstrained per-source-block matrix assigns each of the n blocks a free distribution over 2^m output patterns, $2^m - 1$ parameters each. At $m = 50$, $n = 2$: 100 against $\approx 2.25 \times 10^{15}$. □

The bound is tight: m queries with independently chosen rates $(\varepsilon_\ell, \omega_\ell)$ satisfy both axioms, and distinct parameter tuples give distinct joint matrices, so no reparameterization beats $2m$. The companion paper [Tow26b]

records the factorization as a remark in the deterministic-or-conditioned regime; the explicit conditional form, the distinct-block unconditional corollary, and the Jensen-gap accounting are this book's contribution.

The BSC, queried m times

The binary symmetric channel of Section 2.1 is the cleanest instance. It has one crossover parameter $\varepsilon = \omega$, so its two inputs share a rate and merge into a single block: the BSC is model order 1 (Section 2.4), studied as the symmetric constrained member of the order-2 family. In the deterministic-rates regime, m independent queries through it have joint matrix $Q^{\otimes m}$. For $m = 2$,

$$Q^{\otimes 2} = \begin{pmatrix} (1-\varepsilon)^2 & (1-\varepsilon)\varepsilon & \varepsilon(1-\varepsilon) & \varepsilon^2 \\ (1-\varepsilon)\varepsilon & (1-\varepsilon)^2 & \varepsilon^2 & \varepsilon(1-\varepsilon) \\ \varepsilon(1-\varepsilon) & \varepsilon^2 & (1-\varepsilon)^2 & (1-\varepsilon)\varepsilon \\ \varepsilon^2 & \varepsilon(1-\varepsilon) & (1-\varepsilon)\varepsilon & (1-\varepsilon)^2 \end{pmatrix},$$

each entry a product of two transition probabilities. The general order-2 channel $Q = \left(\begin{smallmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{smallmatrix}\right)$ verifies the same way: the $(B_1B_1, ++)$ entry of $Q \otimes Q$ is τ^2, which is $\mathrm{P}(\text{both } \top \mid \text{both positive}) = \tau \cdot \tau$ by Axiom 1, with each τ the same stable value by Axiom 2. The two-query joint thus carries 4 parameters, not the $2(2^2 - 1) = 6$ a general channel would.

This is the theorem the AND derivation of Section 3.1 tacitly used: the two operands' joint channel is $Q_1 \otimes Q_2$, and the AND output is a deterministic function of that joint, with Axiom 1 supplying the factorization and Axiom 2 the stable rates. The serial composition theorem (matrix product) and this Kronecker factorization (tensor product) are sister consequences of the same two axioms in different geometries: serial composition threads one element through a chain, the Kronecker product queries m elements in parallel. With the theorem in hand, the formal definition of Section 4.5 is parametrized by finitely many numbers, two in the Boolean case, rather than an exponential family of joint distributions.

4.5 The Bernoulli[bool] Type, Formally

The two axioms and the Kronecker theorem earn the formal definition the first three chapters circled.

Definition 4.5.1 (Bernoulli[bool])**.** *A* Bernoulli[bool] instance *is a triple* (X, Y, Q) *with input and output domains* $X = Y = \{\top, \bot\}$*, blocks* $B_1 = \{\top\}$

and $B_2 = \{\bot\}$*, and a* 2×2 *row-stochastic channel matrix*

$$Q \;=\; \begin{pmatrix} \tau & \omega \\ \varepsilon & \eta \end{pmatrix} \;=\; \begin{pmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{pmatrix},$$

row i *governing block* B_i*, with* ε *the false-positive rate,* ω *the false-negative rate, and* $\tau = 1-\omega$*,* $\eta = 1-\varepsilon$*. The triple satisfies Axiom 1 (Axiom 1) and Axiom 2 (Axiom 2).*

We write Bernoulli[bool] for the type, the collection of all such triples; a predicate or data structure is an *instance* when its behavior matches the definition. The pair (ε, ω) identifies Q within the type. Axiom 1 is what lets Q be read as a per-element description (without it, cross-input correlation needs a richer object), and Axiom 2 is what makes Q time-invariant; together they are the content of Corollary 4.4.1.2. The definition is for the Boolean case; the extension to Bernoulli[T] for general T is Part III.

Example 1 (Biased coin) *A coin lands heads (*$\top$*) with prior* p*, observed through a channel with rates* (ε, ω)*. The joint distribution over (truth, report) pairs is* $p\,\tau$*,* $p\,\omega$*,* $(1-p)\,\varepsilon$*,* $(1-p)\,\eta$ *for TP, FN, FP, TN, summing to one. The type characterizes the channel, not the source: the prior* p *is not part of* (X, Y, Q)*, though it enters when computing outcome probabilities. At* $p = 0.6$*,* $\varepsilon = 0.05$*,* $\omega = 0.10$ *the four outcomes are* $0.540, 0.060, 0.020, 0.380$*, and* $Q = \left(\begin{smallmatrix} 0.90 & 0.10 \\ 0.05 & 0.95 \end{smallmatrix}\right)$ *is a well-formed* Bernoulli[bool] *instance. Axiom 1 is vacuous for a single query; Axiom 2 is consistent with choosing* ε *and* ω *without coupling.*

Chapters 1 through 3 as instances

The definition names what the first three chapters were doing. The three faces of Section 1.1 are all Bernoulli[bool]: the biased coin above, the binary symmetric channel of Section 2.1 with $\varepsilon = \omega$, and the classifier verdict with rates set by its decision boundary. The asymmetric channel of Section 2.2 is the general Q; the Z-channel and the Bloom filter are the one-sided instance $\omega = 0$.

The gates of chapter 3 are operations on the type. NOT is the unary Bernoulli[bool] $\to$ Bernoulli[bool] that swaps (ε, ω). AND and OR take two instances to one, but with a caveat the formal definition makes precise: the AND output's false-negative rate $1 - (1-\omega_1)(1-\omega_2)$ is prior-free, a fixed type parameter, while its false-positive rate is prior-dependent (the product

$\varepsilon_1\varepsilon_2$ in the all-negative regime, boundary-dependent otherwise, Section 3.1). AND is therefore a clean Bernoulli[bool]-valued operation on its prior-free axis and is type-valued on its prior-dependent axis only relative to a fixed input distribution, the one-sided and Miller-Rabin regimes being where the product holds exactly; OR is dual. The composition theorem of Section 3.3 (serial, matrix product) and the Kronecker factorization (parallel, tensor product) are the two faces of the same independence structure, sister consequences of the axioms rather than one a case of the other.

The atom-outward order was deliberate: examples first, axioms next, the formal definition last, so that it arrives already supported by intuition. The reader who followed chapters 1 through 3 has watched it emerge. Section 4.6 looks ahead to Part II.

4.6 What Comes Next

Part I is complete. The two axioms are stated (Section 4.2, Section 4.3), the Kronecker theorem shows what they license (Theorem 4.4.1), and Definition 4.5.1 gives Bernoulli[bool] a rigorous identity, of which every example in chapters 1 through 3 is an instance. Three of the framework's organizing principles are now in view: the two axioms, the closure of the type under composition (serial and parallel), and the Boolean thread that ran from the noisy bit to the formal definition without leaving $\{\top, \bot\}$. Two more arrive in Part II: parametric parsimony, whose information-theoretic necessity (why $2m$ is the right count, not merely a convenient one) needs the entropy bounds of chapter 8, and space-accuracy duality, the trade-off between a structure's size and its error rates, which is Part II's central concern.

Part II asks what a set-membership predicate built from atoms looks like. A set S over a universe induces a Boolean query per element; an approximate set makes each query a Bernoulli[bool] instance, mutually independent by Axiom 1. The canonical construction is the Bloom filter, a mechanism for manufacturing one-sided instances ($\omega = 0$) with a tunable false-positive rate, developed in Chapter 5; chapter 6 then propagates error through set operations, chapter 7 develops the classification measures, and chapter 8 derives the lower bounds that make space-accuracy duality precise. The Boolean thread becomes the foundation, one Bernoulli[bool] instance at a time.

Bibliographic Notes

The Kronecker product. The product carrying Kronecker's name spread through matrix algebra over the century after his 1866 work on bilinear forms; the standard graduate reference for its algebraic properties, including the mixed-product and vectorization identities, is Horn and Johnson [HJ91, Ch. 4]. The theorem of Section 4.4, that under Axioms 1 and 2 the joint confusion matrix factors as a Kronecker product of per-query matrices, is this work's contribution; the honesty note in that section separates what is new from what descends from the companion papers.

Information-theoretic channels. The proof uses the language of channels and conditional distributions without developing information theory; Cover and Thomas [CT06, Ch. 7–8, 10] covers binary channels, capacity, and the coding theorems at the depth Part I borrows. The Bernoulli axioms are exactly what make the channel memoryless in that vocabulary.

The Bernoulli papers. The two-axiom framework comes from Towell [Tow26b]; Axioms 1 and 2 here are reformulations separated into named sections, each with its own examples, with the Kronecker factorization proved as an explicit consequence rather than a remark. That paper presents the factorization for deterministic or implicitly conditioned rates; this work makes the conditional and unconditional forms explicit (Theorem 4.4.1, Corollary 4.4.1.1), noting that the unconditional form needs deterministic rates or distinct queried blocks. The model-order analysis, that the input-domain partition governs the parameter count, is from Towell [Tow26j], whose general algebraic-type theory has the order-2 Boolean case as its simplest instance. The explicit theorem statement, the constructive proof, and the model-order-to- parameter-count connection are contributions of this monograph.

Part II

From Bits to Sets

Chapter 5

Approximate Sets and Predicates

5.1 From Atom to Set

Part I studied a single noisy bit. Part II takes many of them, arranges them into a structure, and studies the algebra of that structure. The first such structure is the approximate set, and it is forced rather than invented.

A classical set $A \subseteq U$ is characterized entirely by its membership predicate $\mathbb{1}_A : U \to \{\top, \bot\}$, which returns $\top$ on members and $\bot$ elsewhere; two sets are equal exactly when their predicates agree everywhere. The predicate *is* the set, and set-builder notation $\{x : \mathbb{1}_A(x) = \top\}$ is the predicate read backwards. Replace the exact predicate with a noisy one and each query $\mathsf{A}^{\pm}(x)$ becomes an instance of the formal type Bernoulli[bool] of Definition 4.5.1: a Boolean that is usually, but not always, the correct verdict. There is nowhere else for the noise to live, because the membership test is the only behavioral handle on a set.

Definition 5.1.1 (Approximate set)**.** *Let U be a fixed universe and $A \subseteq U$ an* objective set*. An* approximate set *of A is a function $\mathsf{A}^{\pm} : U \to$* Bernoulli[bool] *whose evaluation $\mathsf{A}^{\pm}(x) \in \{\top, \bot\}$ is the structure's report on whether $x \in A$.*

The objective set A is the truth; $\mathsf{A}^{\pm}$ is its noisy report. The model never loses A: the rates of $\mathsf{A}^{\pm}$ are defined against it. What a query loses is the ability to read A off directly. The structure may be a Bloom filter, a perfect hash filter, or a thresholded classifier; what makes it an approximate set is that each query is governed by one 2×2 channel matrix, two rates of error

and nothing else.

The two rates. Every query is one of two kinds. For $x \notin A$ the correct verdict is $\bot$, and the only error is reporting $\top$; for $x \in A$ the correct verdict is $\top$, and the only error is reporting $\bot$. These define the false-positive and false-negative rates,

$$\varepsilon = \mathrm{P}\left(\mathsf{A}^{\pm}(x) = \top | x \notin A\right), \qquad \omega = \mathrm{P}\left(\mathsf{A}^{\pm}(x) = \bot | x \in A\right),$$

the same two rates a single noisy bit carried in Section 1.1, now read at the level of a set. The description does not grow with the set: one ε governs every non-member query and one ω every member query, no matter how large $|A|$ or $|U|$.

The block structure. That collapse from $2|U|$ potential per-element rates to two is not automatic; it is Axiom 2 (Axiom 2) at work. Membership partitions the universe into the in-set block $B_1 = A$ and the out-of-set block $B_2 = U \setminus A$, and the axiom forces one rate per block. An approximate set is thus a Bernoulli[bool] instance whose input domain is U rather than $\{\top, \bot\}$, with the membership partition standing in for the Boolean one. The governing matrix is the chapter-4 channel,

$$Q = \begin{pmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{pmatrix},$$

row 1 for the in-set block, row 2 for the out-of-set block. Many queries inherit the chapter-4 machinery wholesale: Theorem 4.4.1 factors the joint behavior of n distinct queries as the Kronecker product of their per-query channels, using Axiom 1 (Axiom 1) for cross-element independence and Axiom 2 for the shared per-block rate. The full joint description of an n-query workload costs two numbers, not 2^n. Section 5.3 makes the transfer precise; Section 5.2 first builds the canonical instance, the Bloom filter, in closed form, and Section 5.4 draws the line between approximate sets with one structural zero rate and those with two positive rates.

5.2 The Bloom Filter From Scratch

Section 5.1 defined the approximate set abstractly and said nothing about building one in finite space. The Bloom filter is the canonical construction: a bit array plus k hash functions, with a false-negative rate that is structurally zero, a false-positive rate in closed form, and a design knob whose optimal setting we derive. Every later chapter of Part II returns to it.

The construction

Fix a universe U, a bit count m, and a hash count k. The structure is a bit array $b[1\,..\,m]$, initialized to zero, with k hash functions $h_1, \ldots, h_k : U \to \{1, \ldots, m\}$. The analysis treats each h_i as a uniform random function and the k of them as mutually independent; real implementations approximate this with families such as double hashing over a cryptographic digest, and the divergence is small enough to ignore here.

Definition 5.2.1 (Bloom filter)**.** *The* Bloom filter *of parameters* (m, k) *is the structure* $(b, h_1, \ldots, h_k)$ *with two operations:*

- INSERT(x)*: for* $i = 1, \ldots, k$*, set* $b[h_i(x)] \leftarrow 1$*.*
- QUERY(x)*: return* $\top$ *if* $b[h_i(x)] = 1$ *for every* i*; return* $\bot$ *otherwise.*

After inserting the n *elements of* $A \subseteq U$*, the predicate* $x \mapsto$ QUERY(x) *is the membership report of* $\mathsf{A}^{\pm}$*.*

Both operations run in $O(k)$ time, independent of $|A|$ and $|U|$, and the array is the only state. We follow the literature's (m, k, n) convention, with $n = |A|$ the item count; where a later chapter risks confusing this with a workload size we write n_{items}.

Algorithm 2: Bloom filter insert and query

```
1 procedure INSERT(x)
2 |   for i ← 1 to k do
3 |   |   b[h_i(x)] ← 1;
4 |   end
5 function QUERY(x)
6 |   for i ← 1 to k do
7 |   |   if b[h_i(x)] = 0 then return ⊥;
8 |   end
9 |   return ⊤;
```

The false-negative rate is structurally zero

If $x \in A$ was inserted, the bits $b[h_1(x)], \ldots, b[h_k(x)]$ were set to 1 at insertion, and no operation ever resets a bit. Every later QUERY(x) therefore reads k ones and returns $\top$. The argument is structural: it uses only the

monotonicity of the array, not the hash family, the load, or the workload. So

$$\omega \;=\; \mathrm{P}\left(\textsc{Query}(x) = \bot \mid x \in A\right) \;=\; 0$$

for every parameter setting. A Bloom filter is *one-sided* (Section 5.4): one of its two rates is driven to zero by construction rather than by tuning. The remaining question is the size of the other.

The false-positive rate

A query for $x \notin A$ returns $\top$ iff all k bits $b[h_i(x)]$ are 1. Fix a position j. Each of the kn hash values produced during insertion misses j with probability $1 - 1/m$, so

$$\mathrm{P}\left(b[j] = 0\right) \;=\; \left(1 - \tfrac{1}{m}\right)^{kn} \;\approx\; e^{-kn/m},$$

using $\ln(1 - 1/m) \approx -1/m$, with error of order $1/m^2$ per insertion. Write $p = 1 - e^{-kn/m}$ for the resulting bit-occupancy probability; the load kn/m is the average number of hash hits per bit. A query reads k bits, each 1 with marginal probability p. Treating the k reads as independent gives the working formula

$$\varepsilon \;\approx\; \left(1 - e^{-kn/m}\right)^{k}. \tag{5.1}$$

Remark (The bit-independence approximation)**.** *The k queried bits lie in the same array populated by the same n insertions, so their occupancy events are not independent: a collision $h_i(x) = h_{i'}(x)$ between two query hashes makes two of the reads literally the same event, and conditioning on one queried bit being set raises the posterior on the others. The dependence is positive and small, so* (5.1) *slightly* under*-estimates the true rate. The smallest witness takes $m = 4$, $n = 1$, $k = 2$: the inserted element sets one bit with probability $\frac{1}{4}$ (when its two hashes collide) and two bits otherwise, and averaging $\left(\frac{|S|}{4}\right)^2$ over those cases gives an exact false-positive rate of $\frac{1}{4} \cdot \frac{1}{16} + \frac{3}{4} \cdot \frac{1}{4} = \frac{13}{64} \approx 0.203$, against $(1 - e^{-1/2})^2 \approx 0.155$ from* (5.1)*. Both the gap and the collision probability vanish as m grows; at the practical load $kn/m \approx \ln 2$ derived below the deviation is well under a percent of the rate, and we use* (5.1) *throughout.* △

Optimal k for fixed m and n

The bit count m is the space budget and n is fixed by the application; the hash count k is the design knob. Let $f(k) = (1 - e^{-kn/m})^k$, minimized over

$k > 0$ as a continuous variable. Taking logarithms and differentiating,

$$\frac{d}{dk} \ln f(k) \;=\; \ln(1 - e^{-kn/m}) \;+\; \frac{kn}{m} \cdot \frac{e^{-kn/m}}{1 - e^{-kn/m}}.$$

Writing $u = e^{-kn/m}$, so that $kn/m = -\ln u$, the first-order condition $\ln(1-u) - (\ln u)\,\frac{u}{1-u} = 0$ becomes, after multiplying by $(1-u)$,

$$(1-u)\ln(1-u) \;=\; u \ln u. \tag{5.2}$$

By inspection $u = \frac{1}{2}$ solves (5.2), both sides equal $-\frac{1}{2}\ln 2$. It is the only interior solution. Set $\phi(u) = (1-u)\ln(1-u) - u\ln u$ on $(0,1)$; it is antisymmetric about $\frac{1}{2}$, $\phi(1-u) = -\phi(u)$, so $\phi(\frac{1}{2}) = 0$ and $\phi \to 0$ at both endpoints. Its derivative $\phi'(u) = -\ln\big(u(1-u)\big) - 2$ is symmetric about $\frac{1}{2}$, diverges to $+\infty$ at the endpoints, and dips to $\phi'(\frac{1}{2}) = 2\ln 2 - 2 < 0$, so ϕ' changes sign exactly twice. Hence $\phi > 0$ on $(0, \frac{1}{2})$ and $\phi < 0$ on $(\frac{1}{2}, 1)$, vanishing only at $u = \frac{1}{2}$. The boundary behavior pins this critical point as the minimum: $f \in (0,1)$ in the interior and $f \to 1$ at both endpoints (as $k \to 0^+$, $\ln f \approx k\ln(kn/m) \to 0$; as $k \to \infty$, $\ln f \approx -k\,e^{-kn/m} \to 0$), and the second-derivative check at $u = \frac{1}{2}$ gives $\frac{d^2 \ln f}{dk^2} = 2(n/m)(1 - \ln 2) > 0$. Substituting $u = \frac{1}{2}$ into $kn/m = -\ln u$,

$$k^* \;=\; \frac{m}{n} \ln 2, \tag{5.3}$$

at which each bit is occupied with probability exactly $\frac{1}{2}$ and

$$\varepsilon^* \;\approx\; \left(\tfrac{1}{2}\right)^{k^*} \;=\; 2^{-(m/n)\ln 2} \;=\; (0.6185\ldots)^{m/n}. \tag{5.4}$$

Theorem 5.2.1 (Optimal k for the Bloom filter)**.** *For fixed m and n, the false-positive rate* (5.1) *is minimized at $k^* = (m/n)\ln 2$, where the array is half-occupied in expectation and $\varepsilon^* \approx (1/2)^{k^*} = 2^{-(m/n)\ln 2}$.*

The bit-per-item ratio m/n is the right measure of a Bloom filter's space budget: at the optimal k the rate decays exponentially in m/n with base 0.6185, half a bit of accuracy bought per 1.6 bits of storage. Chapter 8 compares this to the information-theoretic floor and finds a fixed gap of $\log_2 e \approx 1.44$; the half-full array at the optimum is the striking signature of the construction, most accurate not when sparse but when exactly half its bits are set.

A worked example

Take $m = 10000$, $n = 1000$, a comfortable $m/n = 10$ bits per item. Then $k^* = 10 \ln 2 \approx 6.93$, rounded to $k = 7$, giving $\varepsilon^* \approx (1/2)^7 = 1/128 \approx 0.0078$, about one false positive per 128 negative queries. Evaluating (5.1) around the optimum confirms the rounding and shows how shallow the minimum is:

k	$1 - e^{-kn/m}$	$\varepsilon \approx (1 - e^{-kn/m})^k$
5	0.3935	0.00943
6	0.4512	0.00844
7	0.5034	0.00819
8	0.5507	0.00846
9	0.5934	0.00913

The rates at $k = 6, 7, 8$ agree to the third significant figure, and the occupancy at $k = 7$ is 0.5034, half-full as predicted. The minimum sits where two pressures balance: raising k widens the conjunction the query must satisfy (pushing the rate down) but fills the array faster (pushing it up), and the two cancel at half occupancy. Off the optimum the penalty is real, $k = 1$ at $m/n = 10$ gives $\varepsilon \approx 0.095$, twelve times the optimum, and the rate depends on m and n only through m/n once k is optimized.

The Bloom filter is one row of the general predicate algebra: $\omega = 0$ structurally, ε in closed form, the same Bernoulli[bool] channel and chapter-4 axioms as any other approximate set. Section 5.3 develops that algebra in generality; Section 5.4 takes up the one-sided property $\omega = 0$ this construction enjoys.

5.3 The Predicate Algebra

Section 5.1 described an approximate set's rates informally and Section 5.2 built one instance. This section places the abstraction inside the chapter-4 machinery at the matrix level: the per-element channel, the Axiom-2 collapse to two per-block rates, and the Kronecker factorization over many queries. The payoff is the chapter's first formal statement of parametric parsimony, that an approximate set's entire joint behavior is fixed by two numbers.

Per-element channels and their collapse

For each $x \in U$ the query $\mathsf{A}^{\pm}(x)$ is a Bernoulli[bool] instance. In the most general setting each x carries its own rate, $\omega(x)$ if $x \in A$ and $\varepsilon(x)$ if $x \notin A$,

assembling into a per-element channel

$$Q_x \;=\; \begin{pmatrix} 1-\omega(x) & \omega(x) \\ \varepsilon(x) & 1-\varepsilon(x) \end{pmatrix},$$

row 1 for the in-set case, row 2 for the out-of-set case, matching the convention of Definition 4.5.1. Unconstrained, the family $\{Q_x\}$ carries up to $2|U|$ parameters. Axiom 2 collapses it.

Proposition 5.3.1 (Block-stability of per-element rates)**.** *Let* $\mathsf{A}^{\pm} : U \to$ Bernoulli[bool] *satisfy Axiom 2 (Axiom 2) over the partition* $\{A, U \setminus A\}$*. Then there exist scalars* $\varepsilon, \omega \in [0,1]$ *with* $\omega(x) = \omega$ *for all* $x \in A$ *and* $\varepsilon(x) = \varepsilon$ *for all* $x \notin A$*, so the per-element channel is one fixed matrix* Q *for every query, and the rate description collapses from* $2|U|$ *parameters to the pair* (ε, ω)*.*

Proof. Axiom 2 assigns each block B_i a single rate α_i. Proposition 4.3.1 then makes Q_x depend on x only through its block, with row i determined by α_i. For the two-block partition this gives ω on row 1 (every $x \in A$) and ε on row 2 (every $x \notin A$); the off-diagonal entries follow from row-stochasticity. Since the block fixes the row and the two rows are the whole matrix, $Q_x = Q$ for every x. □

Throughout Part II we work in the deterministic-rates regime of Chapter 4, where ε and ω are fixed parameters rather than random variables; the conditional and unconditional Kronecker forms then coincide and we write a single Q for $\mathsf{A}^{\pm}$. What varies across queries is the row a query reads, not the matrix.

Remark (Heterogeneous blocks)**.** *The collapse is exactly as strong as Axiom 2. A construction whose error rate genuinely varies* within *a block escapes it. A learned classifier used as a membership oracle might answer far-from-boundary out-of-set queries at* $\varepsilon_{easy} = 0.01$ *but near-boundary ones at* $\varepsilon_{hard} = 0.30$*, so* $\varepsilon(x)$ *depends on* x *inside* B_2 *and no single block rate fits. Corollary 5.3.0.1 below then fails in its two-parameter form: the joint distribution must carry per-element rates and is no longer universe-size-independent. The gap is workload-dependent. A workload sampling uniformly from* B_2 *averages over the heterogeneity and looks single-rate; an adversarial workload concentrating on near-boundary elements exposes it. This is the boundary between the clean theory and the classifiers of Chapter 7.* △

The joint distribution factors via Kronecker

A workload of m queries on distinct elements $x_1, \ldots, x_m$, with x_ℓ in block j_ℓ and report k_ℓ, has joint behavior governed by Theorem 4.4.1. Its hypotheses are the two axioms: Axiom 1 (Axiom 1) for cross-element independence and Axiom 2 for the block-stability just established. Since every per-query matrix is the same Q, the joint confusion matrix is the m-fold Kronecker power.

Corollary 5.3.0.1 (Kronecker factorization for approximate sets)**.** *Let* $\mathsf{A}^{\pm}$ *satisfy Axioms 1 and 2 over* $\{A, U \setminus A\}$ *in the deterministic-rates regime. The joint confusion matrix of* m *queries on* m *distinct elements is*

$$Q^{(m)} \;=\; Q^{\otimes m}, \qquad Q \;=\; \begin{pmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{pmatrix},$$

so the full joint description of $\mathsf{A}^{\pm}$ *requires only* (ε, ω)*, independent of* $|U|$ *and* $|A|$*.*

Proof. Proposition 5.3.1 makes every per-element matrix the same Q. Theorem 4.4.1 factors $Q^{(m)} = Q_1 \otimes \cdots \otimes Q_m$, and in the deterministic-rates regime the conditional and unconditional forms agree with every factor equal to Q, so $Q^{(m)} = Q^{\otimes m}$. The two scalars fixing Q fix the whole power. □

This earns at the joint level the informal claim of Section 5.1: two scalars fix Q, Q fixes the Kronecker power, and the power fixes the behavior of arbitrarily many queries. To see it concretely, take $U = \{1, \ldots, 10\}$, $A = \{1, 2, 3\}$, $\varepsilon = 0.1$, $\omega = 0.05$, so $Q = \left(\begin{smallmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{smallmatrix}\right)$. The probability that two in-set queries and two out-of-set queries all report $\top$ (two true positives, two false positives) is the product of the corresponding cells,

$$Q_{1,1}^2 \, Q_{2,1}^2 \;=\; (1-\omega)^2 \varepsilon^2 \;=\; 0.95^2 \cdot 0.10^2 \;=\; 0.009025,$$

while the all-correct outcome has probability $(1-\omega)^2(1-\varepsilon)^2 = 0.731$. Doubling the universe leaves these unchanged; doubling the workload grows the Kronecker power but keeps each entry a product of two-rate factors, one per query. The Bloom filter is the special case $\omega = 0$; any other construction supplies a different Q but the same algebra. The distinction Section 5.4 draws next is what happens when one of the two parameters is structurally zero.

5.4 One-Sided vs Two-Sided Approximate Sets

The Bloom filter has one error rate exactly zero, by construction rather than tuning. That is one shape an approximate set can take; others force the false-positive rate to zero on a designated set, or allow both rates to be positive. The shapes are not interchangeable, and the distinction organizes the constructions of Chapter 6, Chapter 14, and Chapter 15.

Definition 5.4.1 (One-sided and two-sided approximate sets)**.** *An approximate set* $\mathsf{A}^{\pm}$ *with rates* (ε, ω) *is* one-sided in FNR *(FNR-zero) if* $\omega = 0 < \varepsilon$*;* one-sided in FPR *(FPR-zero) if* $\varepsilon = 0 < \omega$*; and* two-sided *if both rates are positive. The Bloom filter (Definition 5.2.1) is FNR-zero; the perfect hash filter of Chapter 15 is FPR-zero on its design domain.*

The name tracks the rate that is zero. A one-sided set sits on a coordinate axis of the rate square $[0, 1]^2$; a two-sided set in its interior; the origin $\varepsilon = \omega = 0$ is the exact set, excluded as not approximate. The Z-channel of Definition 2.2.2 is the per-query analog, and the one-sided property is its per-set lift.

Why FNR-zero is the natural default

The Bloom filter's FNR-zero property came for free from monotonicity: the insert primitive only sets bits, so a query for an inserted item always reads ones. Any structure on a monotone insert-only primitive shares it, which is why summaries over append-only logs and hash-keyed sketches are FNR-zero for the same structural reason. Forcing the opposite asymmetry is harder. Making $\varepsilon = 0$ on a designated set requires the construction to certify membership, which a stored copy of the set would do, at the cost of the space an approximate set exists to save. The constructive path must be cleverer: Chapter 15 computes a perfect hash for a fixed A and stores one fingerprint per element, so a query for $x \in A$ lands on its fingerprint and is certain while a query for $x \notin A$ may still collide. The set must be known up front and updates are expensive; those costs buy certainty in the positive direction.

Safety versus space

One-sidedness is valuable because one direction of the answer is exact: a Bloom filter's $\perp$ is certain (the element is definitely absent), so a cache can skip a lookup or a spell-checker can pass a known word without follow-up. An FPR-zero filter has the dual guarantee, a certain $\top$. Two-sided

sets give up the certain direction to recover space, as a *counting* Bloom filter does: it replaces bits with small counters to support deletion, and counter saturation, where an overflowed counter is later decremented to zero, introduces a positive ω along with a roughly fourfold space cost.

The size of the trade is calculable. By Equation (5.3) a Bloom filter at $\varepsilon = 0.01$, $\omega = 0$ needs $m \approx 1.44\, n \log_2(1/\varepsilon) \approx 9.6\, n$ bits. Chapter 8 shows the floor for any FNR-zero set at the same ε is $n \log_2(1/\varepsilon) \approx 6.64\, n$ bits, so the Bloom filter pays a fixed $1.44\times$ overhead, the standard literature's $\log_2 e$ constant, which Chapter 14 closes. A two-sided set is not bound by the one-sided floor at ε, because it can split its error budget across both directions, but it spends the certain $\perp$ that the extra bits bought.

The perfect filter line

The rate plane $(\varepsilon, \omega) \in [0, 1]^2$ parameterizes every approximate set. The one-sided sets occupy the two axes, $\{\omega = 0\}$ and $\{\varepsilon = 0\}$; we call their union the *perfect filter line*, each axis a one-dimensional locus with one rate-degree of freedom. Two-sided sets fill the interior; the exact set is the origin.

The line is an algebraic constraint, not a passive label. Chapter 6 derives the union, intersection, and complement of Bernoulli sets and shows:

- the union of two FNR-zero sets is FNR-zero;
- the intersection of two FNR-zero sets is FNR-zero;
- the complement swaps the axes, sending FNR-zero to FPR-zero.

The FPR-zero side is not closed under the same operations: the union of two FPR-zero sets is FPR-zero only when both are exact on the same designated set, a fragile condition. The asymmetry parallels the construction-side one, FNR-zero the natural default, FPR-zero requiring care. Section 3.5 already previewed the union-as-bit-OR identity for FNR-zero filters as a special case of the composition theorem; Chapter 6 generalizes it. Chapter 14 and Chapter 15 then take up the two halves of the line: the FNR-zero family and its entropy floor, and the FPR-zero construction that pays the dual price.

5.5 What Comes Next

The chapter lifted the noisy bit to a function $\mathsf{A}^{\pm} : U \to \text{Bernoulli[bool]}$ (Section 5.1), built the Bloom filter as its canonical realization (Section 5.2), placed both inside the chapter-4 axioms so that two rates fix the entire

joint behavior (Section 5.3), and separated the rate shapes an approximate set can take (Section 5.4). The rest of Part II develops the algebra and bounds the construction underwrites. Chapter 6 derives the rates of unions, intersections, complements, and differences as the composition theorem of Section 3.3 applied to the predicate algebra, and asks which subfamilies, the perfect filter line among them, are closed under each operation. Chapter 7 turns the two intrinsic rates into the workload-dependent measures practitioners report, positive and negative predictive value, recall, F_1, and supplies the proper interval estimators that Section 1.4 deferred. Chapter 8 proves the entropy floor of $-\log_2 \varepsilon$ bits per element for an FNR-zero set and locates the standard Bloom filter's $1.44\times$ overhead against it. The two constructions that take the perfect filter line as their subject, Chapter 14 closing the FNR-zero gap and Chapter 15 realizing the FPR-zero side, arrive in Parts IV and V once the intervening theory is in place.

Bibliographic Notes

Bloom's original paper. The construction is due to Bloom [Blo70], a four-page 1970 note framed operationally: hyphenation dictionaries for typesetting, trading a small rate of mishyphenated words for not storing the dictionary. That paper already contains the optimal-k derivation Theorem 5.2.1 reproduces, including $k = (m/n)\ln 2$ and the per-bit rate $(1/2)^k$ at the optimum. What this monograph adds is the framing, the Bloom filter as one construction of a function $U \to$ Bernoulli[bool] rather than a bit-array trick; the two readings agree on every numerical prediction.

Survey and textbook context. Mitzenmacher and Upfal [MU17] treat hashing, Bloom filters, and the balls-and-bins analysis underlying both, with the same optimal-k calculus used in Section 5.2 but without the Bernoulli[bool] framing. Broder and Mitzenmacher [BM04] survey applications across networking, distributed systems, and databases, and document the variants, counting, scalable, and compressed filters, that the present chapter mentions only in passing.

The set-as-predicate viewpoint. Reading an approximate set as a function $U \to$ Bernoulli[bool], rather than a data structure that happens to have an error rate, is the contribution of Towell [Tow26b]. The earlier literature, including Bloom [Blo70] and Broder and Mitzenmacher [BM04], characterizes such a structure by a single false-positive probability; the Bernoulli

reading instead makes the function the object of study, its per-query output a Bernoulli[bool] instance in the sense of Towell [Tow26j], and the data structure one realization among many. The algebra of Section 5.3 is the direct lift of the order-2 algebra of chapter 4 to the per-element setting.

One-sided and perfect-filter lineage. The dichotomy of Definition 5.4.1 has two ancestors. The FNR-zero shape is the Bloom filter's structural property, named in Bloom [Blo70] but isolated as a category only once the literature accumulated several constructions sharing it. The FPR-zero shape, a perfect filter on a designated set, has constructive roots in the universal hashing of Carter and Wegman [CW79]: without a perfect hash on A, a fingerprint table cannot guarantee $\varepsilon = 0$ on $x \in A$. The perfect hash filter of Chapter 15 is the modern descendant. Both Chapter 14 and Chapter 15 build on Towell [Tow26m], which derives the space-optimal Bernoulli hash filter for the unrestricted and FPR-zero regimes and proves the entropy lower bound that makes "optimal" precise.

Chapter 6

Composition of Sets

6.1 Union of Approximate Sets

Chapter 5 framed an approximate set as a noisy membership predicate with two per-block rates (ε, ω). The first compositional step asks what the union of two such sets is and what rates it inherits. The answer is the composition theorem of Theorem 3.3.1 applied to the OR gate of Section 3.2, lifted from the per-bit level to the per-set level.

Definition 6.1.1 (Union of approximate sets)**.** *For approximate sets* $\mathsf{A}^{\pm}, \mathsf{B}^{\pm} : U \to$ Bernoulli[bool] *over a common universe, their* union $\mathsf{A}^{\pm} \cup \mathsf{B}^{\pm}$ *is the approximate set whose predicate is the Boolean OR of the operand predicates,* $(\mathsf{A}^{\pm} \cup \mathsf{B}^{\pm})(x) = \mathsf{A}^{\pm}(x) \vee \mathsf{B}^{\pm}(x)$*, approximating the objective set* $A \cup B$*.*

To evaluate the union at x, query both operands and OR the verdicts; the rates follow from the operand rates and the OR-gate analysis of Chapter 3.

Proposition 6.1.1 (Union rate formulas)**.** *Let* $\mathsf{A}^{\pm}, \mathsf{B}^{\pm}$ *approximate* $A, B \subseteq U$ *with rates* $(\varepsilon_A, \omega_A)$*,* $(\varepsilon_B, \omega_B)$*, and assume the operand error events at each query point are mutually independent. Then the union has false-positive rate*

$$\varepsilon_{\cup} \;=\; 1 - (1 - \varepsilon_A)(1 - \varepsilon_B), \tag{6.1}$$

and, for a query distribution drawing x *from the regions* $A \setminus B$*,* $B \setminus A$*,* $A \cap B$ *of* $A \cup B$ *with weights* w_1, w_2, w_3*, false-negative rate*

$$\omega_{\cup} \;=\; w_1 \omega_A (1 - \varepsilon_B) + w_2 \omega_B (1 - \varepsilon_A) + w_3 \omega_A \omega_B. \tag{6.2}$$

Proof. For $x \notin A \cup B$, the union reports $\top$ iff at least one operand false-positives; the complementary event factors by independence, giving $\varepsilon_{\cup} =$

$1-(1-\varepsilon_A)(1-\varepsilon_B)$, the same for every negative of $A \cup B$. A false negative needs both operands to report $\perp$, and the per-region probability depends on the operands' truth there: on $A \setminus B$ the first must miss a member while the second correctly rejects, $\omega_A(1-\varepsilon_B)$; on $B \setminus A$ symmetrically $\omega_B(1-\varepsilon_A)$; on $A \cap B$ both must miss, $\omega_A\omega_B$. The law of total probability over the three regions gives (6.2). □

The asymmetry between the two rates is structural and recurs for every operation in this chapter. The union FPR is a closed-form polynomial in the operand FPRs alone, requiring no prior, because every negative of $A \cup B$ lies in the single region $U \setminus (A \cup B)$. The union FNR mixes three per-region rates, because the positives of $A \cup B$ split into three blocks where the operands play different roles; the composed set is higher-order, and (6.2) reports its marginal FNR over a query distribution. When the operands are FNR-zero ($\omega_A = \omega_B = 0$) every term vanishes and the union stays FNR-zero, the first closure fact of Section 6.4.

Two realizations on Bloom filters. Definition 6.1.1 is a predicate-level operation, *predicate-OR*: keep both filters, query each, OR the outputs; its FPR is (6.1). When both operands are Bloom filters with the same parameters and hash family, a tempting shortcut is *bit-OR-Bloom*: OR the two bit arrays into one bitmap and query it once. This is a different operation. Bit-OR-Bloom is exactly a Bloom filter holding the union of the inserted items, so its rate is the chapter-5 formula at the combined count,

$$\varepsilon_{\text{bit-OR}} \;\approx\; \left(1 - e^{-k(n_A+n_B)/m}\right)^k, \tag{6.3}$$

materially larger than (6.1): at $m = 5000$, $k = 7$, $n_A = n_B = 500$ the operand rate is $\varepsilon \approx 0.0082$, predicate-OR gives ≈ 0.0163, and bit-OR-Bloom gives ≈ 0.137. Both preserve FNR-zero, but they ask different questions. Predicate-OR requires one filter to have *all* k bits set; bit-OR-Bloom requires only that at each queried position *some* filter's bit is set, with the responsible filter free to vary by position, so it is the more permissive operation and carries the larger FPR. Predicate-OR is the union of Definition 6.1.1; bit-OR-Bloom is a representation choice that trades a higher FPR for one bitmap and one lookup, and it further requires a shared hash family.

For two FNR-zero Bloom filters at $\varepsilon_A = 0.01$, $\varepsilon_B = 0.02$ the predicate-OR union has $\varepsilon_\cup = 1-(1-0.01)(1-0.02) = 0.0298$ and $\omega_\cup = 0$. The second-order term $\varepsilon_A\varepsilon_B = 0.0002$ is small, so for small rates the union FPR is nearly additive, $\varepsilon_\cup \approx \varepsilon_A + \varepsilon_B$, bracketed by $\max(\varepsilon_A, \varepsilon_B) \le \varepsilon_\cup \le \varepsilon_A + \varepsilon_B$.

(6.1) is the composition theorem of Theorem 3.3.1 applied to the OR gate with $\eta_i = \varepsilon_i$, the lift to sets that Section 3.5 previewed: the same Boolean operation, the same algebra, with ε_i reread from a per-bit crossover to a per-block set rate. The n-ary union is the same theorem folded, $\varepsilon_\cup = 1 - \prod_i (1 - \varepsilon_i)$, and since OR is associative and commutative with identity $\varnothing^\pm$, the union is a monoidal fold over approximate sets.

Remark (Cross-set independence)**.** *Proposition 6.1.1 assumes the operand error events are independent at each point, and the conclusion is exactly as good as that assumption. Two Bloom filters built on the* same *hash family and bit array over the same stored set are not independent but identical,* $\mathsf{A}^\pm(x) = \mathsf{B}^\pm(x)$ *everywhere, so their predicate-OR is a single filter with FPR* ε*, not the* $1-(1-\varepsilon)^2 = 2\varepsilon-\varepsilon^2$ *the formula predicts;* (6.1) *over-states the rate. Independence is what the product of marginals encodes, and constructions that share randomness across operands fall outside it, the cross-query analog of the within-query dependence Line 9 flagged for a single filter. This is why Definition 6.1.1 keeps the operands as separate objects.* △

Section 6.2 takes up intersection, complement, and difference; Section 6.3 the regimes in which a composition leaves the perfect filter line; Section 6.4 the line as a formal subalgebra.

6.2 Intersection, Complement, Difference

The union of Section 6.1 lifts the OR gate to sets; the remaining Boolean operations lift the same way. Each composed predicate is a Boolean combination of the operand predicates, and its rates follow from the gate analysis of Chapter 3 and cross-set independence. The recurring structure of Proposition 6.1.1 persists: one rate is a clean product over a homogeneous truth-region, the other is a partition-weighted sum over regions where the operands play different roles.

Intersection

Definition 6.2.1 (Intersection of approximate sets)**.** *For* $\mathsf{A}^\pm, \mathsf{B}^\pm$ *over* U*, the* intersection $\mathsf{A}^\pm \cap \mathsf{B}^\pm$ *has predicate* $(\mathsf{A}^\pm \cap \mathsf{B}^\pm)(x) = \mathsf{A}^\pm(x) \wedge \mathsf{B}^\pm(x)$*, approximating* $A \cap B$*.*

Proposition 6.2.1 (Intersection rate formulas)**.** *Under the cross-set independence of Proposition 6.1.1, the intersection has false-negative rate*

$$\omega_\cap \;=\; 1 - (1 - \omega_A)(1 - \omega_B), \tag{6.4}$$

and, for a query distribution drawing x from the negatives of $A \cap B$ in the regions $A \setminus B$, $B \setminus A$, $U \setminus (A \cup B)$ with weights w_1, w_2, w_3, false-positive rate

$$\varepsilon_{\cap} \;=\; w_1(1-\omega_A)\varepsilon_B + w_2\varepsilon_A(1-\omega_B) + w_3\varepsilon_A\varepsilon_B. \tag{6.5}$$

Proof. For $x \in A \cap B$, the intersection reports $\bot$ (a false negative) iff at least one operand misses; the complementary joint event factors by independence, giving (6.4), the De Morgan dual of the union FPR (6.1). A false positive needs both operands to report $\top$ at an $x \notin A \cap B$, and the per-region probability depends on the operands' truth there: on $A \setminus B$ the first correctly accepts a member while the second false-positives, $(1-\omega_A)\varepsilon_B$; on $B \setminus A$ symmetrically $\varepsilon_A(1-\omega_B)$; on $U \setminus (A \cup B)$ both false-positive, $\varepsilon_A\varepsilon_B$. Total probability over the three regions gives (6.5). □

The FPR is the dual of the union FNR (6.2), with the same three-region structure: the intersection of two approximate sets is a higher-order Bernoulli set, and (6.5) is its marginal FPR. The bare product $\varepsilon_A\varepsilon_B$ is only the third term, the both-outside region. It is exact when the two operands approximate the *same* objective set ($A = B$, so $A \setminus B = B \setminus A = \varnothing$ and $w_1 = w_2 = 0$), and it is the limit when the universe dwarfs both sets ($w_3 \to 1$). Outside those regimes the first two terms dominate, and they are *linear* in a single operand rate, not quadratic: a query landing in $A \setminus B$ false-positives at rate ε_B alone, because $\mathsf{A}^{\pm}$ accepts the genuine member. The widely quoted "intersection multiplies false-positive rates" holds for filters of one set: intersecting k independent FNR-zero filters of a common A drives the FPR to $\prod_i \varepsilon_i \to 0$, since on the dominant outside region a negative survives only if every filter false-positives.

Complement

Definition 6.2.2 (Complement of an approximate set)**.** *The* complement $\complement\mathsf{A}^{\pm}$ *has predicate* $(\complement\mathsf{A}^{\pm})(x) = \neg\mathsf{A}^{\pm}(x)$*, approximating* $\complement A = U \setminus A$*.*

Proposition 6.2.2 (Complement swaps the rates)**.** *If* $\mathsf{A}^{\pm}$ *has rates* $(\varepsilon_A, \omega_A)$*, then* $\complement\mathsf{A}^{\pm}$ *has rates*

$$\varepsilon_{\complement} = \omega_A, \qquad \omega_{\complement} = \varepsilon_A. \tag{6.6}$$

Proof. A false positive of $\complement\mathsf{A}^{\pm}$ is the event $x \in A$ (a negative of $\complement A$) with $\mathsf{A}^{\pm}(x) = \bot$, a false negative of $\mathsf{A}^{\pm}$ at rate ω_A. A false negative is $x \notin A$ with $\mathsf{A}^{\pm}(x) = \top$, a false positive of $\mathsf{A}^{\pm}$ at rate ε_A. The rates swap. □

This is the set-level reading of NOT-under-noise from Section 3.2. Complement is an involution exchanging the two coordinate axes of the rate plane; unlike intersection and difference, it needs no partition, because NOT acts pointwise and reclassifies every element by the same rule.

Difference

Definition 6.2.3 (Difference of approximate sets)**.** *The* difference $\mathsf{A}^{\pm} \setminus \mathsf{B}^{\pm} = \mathsf{A}^{\pm} \cap \complement\mathsf{B}^{\pm}$ *has predicate* $\mathsf{A}^{\pm}(x) \wedge \neg\mathsf{B}^{\pm}(x)$*, approximating* $A \setminus B = A \cap \complement B$*.*

Proposition 6.2.3 (Difference rate formulas)**.** *Under the same independence, the difference has false-negative rate*

$$\omega_{\setminus} \;=\; 1 - (1-\omega_A)(1-\varepsilon_B), \tag{6.7}$$

and, over the negatives of $A \setminus B$ *in the regions* $A \cap B$*,* $U \setminus (A \cup B)$*,* $B \setminus A$ *with weights* w_1, w_2, w_3*, false-positive rate*

$$\varepsilon_{\setminus} \;=\; w_1(1-\omega_A)\omega_B + w_2\varepsilon_A(1-\varepsilon_B) + w_3\varepsilon_A\omega_B. \tag{6.8}$$

Proof. Write $\mathsf{A}^{\pm} \setminus \mathsf{B}^{\pm} = \mathsf{A}^{\pm} \cap \complement\mathsf{B}^{\pm}$ and apply Proposition 6.2.1 with $\complement\mathsf{B}^{\pm}$ carrying rates $(\omega_B, \varepsilon_B)$ by Proposition 6.2.2. The FNR is $1-(1-\omega_A)(1-\varepsilon_B)$, which is (6.7). The FPR substitutes the swapped second-operand rates into the three-region sum, with the regions relabeled for the objective set $A \cap \complement B$: on $A \cap B$, $\mathsf{A}^{\pm}$ accepts and $\complement\mathsf{B}^{\pm}$ false-positives (rate ω_B); on $U \setminus (A \cup B)$, $\mathsf{A}^{\pm}$ false-positives and $\complement\mathsf{B}^{\pm}$ accepts (rate $1-\varepsilon_B$); on $B \setminus A$, $\mathsf{A}^{\pm}$ false-positives and $\complement\mathsf{B}^{\pm}$ false-positives (rate ω_B). This is (6.8). □

Both operands' *both* rates appear in the difference. The consequence is sharp and is the reason the difference matters for the perfect filter line: even an FNR-zero $\mathsf{A}^{\pm}$ inherits false negatives at rate ε_B, because a member of $A \setminus B$ that $\mathsf{B}^{\pm}$ false-positives is wrongly excluded; and even FNR-zero operands leave a positive difference FPR through the $w_2\varepsilon_A(1-\varepsilon_B)$ term, because a point outside both sets that $\mathsf{A}^{\pm}$ false-positives passes the difference (the complemented $\mathsf{B}^{\pm}$ correctly accepts it). The difference of two one-sided sets is therefore two-sided, a fact the worked example makes concrete and Section 6.3 reads as an escape from the line.

Worked example

Take two FNR-zero Bloom filters with $\varepsilon_A = 0.01$, $\omega_A = 0$ and $\varepsilon_B = 0.02$, $\omega_B = 0$, in a universe large enough that the region outside both sets dominates each partition ($w_3 \to 1$ for the intersection, $w_2 \to 1$ for the difference).

Intersection. The FNR is $1-(1-0)(1-0)=0$: the intersection of two FNR-zero sets stays FNR-zero. The FPR is (6.5), dominated by the outside-both term, $\varepsilon_\cap \approx \varepsilon_A\varepsilon_B = 0.01\times0.02 = 0.0002$, an order of magnitude below either operand. (Were the two sets distinct enough that a query could land in $A\setminus B$ or $B\setminus A$, the linear terms $w_1\varepsilon_B + w_2\varepsilon_A$ would lift the rate well above the product.) Intersection of same-set filters drives the FPR down; union inflates it; the two are exact duals.

Complement. $\mathsf{CA}^{\pm}$ has rates $(\omega_A, \varepsilon_A) = (0, 0.01)$: the FNR-zero false-positive machine becomes an FPR-zero false-negative machine, moved from the horizontal axis to the vertical one.

Difference. Here the products mislead. Naively pairing ε_A with $\omega_B = 0$ would suggest $\varepsilon_\setminus = 0$; but that is only the w_3 term of (6.8). The dominant w_2 term survives,

$$\varepsilon_\setminus \approx \varepsilon_A(1-\varepsilon_B) = 0.01\times0.98 = 0.0098, \qquad \omega_\setminus = 1-(1-0)(1-0.02) = 0.02.$$

The difference of two FNR-zero Bloom filters is *two-sided*, at approximately $(0.0098, 0.02)$, not the FPR-zero point $(0, 0.02)$ the bare product would give. Both failure modes are real: $\mathsf{B}^{\pm}$'s false positives become wrongful exclusions (the FNR $= \varepsilon_B$), and $\mathsf{A}^{\pm}$'s false positives on points outside both sets survive the difference (the FPR $\approx \varepsilon_A$). This two-sidedness is the cancellation phenomenon Section 6.3 scrutinizes.

A caveat on bit-AND of Bloom filters

Definition 6.2.1 is a predicate operation: query both filters, AND the verdicts, and Proposition 6.2.1 governs it. The operational shortcut of bit-wise AND of two bit arrays is a different construction. On items inserted into both sets the bit-AND query matches the predicate AND, but on items inserted into one set only it can still report $\top$ when both AND'd bits happen to be set (one legitimate, one a coincidental collision), so the bit-AND is not in general a Bloom filter for $A \cap B$. We treat intersection at the predicate level and do not rely on bit-AND in the rate analyses; the chapter notebook measures the discrepancy.

6.3 Cancellation and the Loss of One-Sidedness

The operands of Section 6.1 and Section 6.2 were drawn from one-sided sets on the perfect filter line. Some compositions kept the result on the operands' axis; others did not. This section classifies the three behaviors: a

composition can stay on the operands' axis, walk to the other axis, or leave the line entirely into the two-sided interior. The complement and the mixed-axis intersection drive the last two, and Section 6.4 rereads the classification as a closure question.

Complement swaps which axis

Proposition 6.3.1 (Complement swaps the perfect-filter axis)**.** *If* $\mathsf{A}^{\pm}$ *is one-sided, so is* $\complement\mathsf{A}^{\pm}$*, on the other axis: FNR-zero maps to FPR-zero and vice versa.*

Proof. By Proposition 6.2.2 the complement sends $(\varepsilon_A, \omega_A)$ to $(\omega_A, \varepsilon_A)$. If $\omega_A = 0 < \varepsilon_A$ then the image is $(0, \varepsilon_A)$ with $\varepsilon_{\complement} = 0 < \omega_{\complement}$, which is FPR-zero. The FPR-zero case is symmetric. □

A Bloom filter at $(0.01, 0)$ on the FNR-zero axis has complement $(0, 0.01)$ on the FPR-zero axis: the same information read dually, certain about absence becoming certain about presence. The complement is a discrete reflection across the rate plane, not a continuous deformation; it walks the line from one arm to the other in a single step. Composing union with complement illustrates this without leaving the line. Two FNR-zero filters at $\varepsilon_A = 0.01$, $\varepsilon_B = 0.02$ have union $(0.0298, 0)$ by Proposition 6.1.1, still FNR-zero, and complementing the union swaps its rates to $(0, 0.0298)$ on the FPR-zero axis. The composed structure stays one-sided; only the certain direction flips, from "is x in A or B?" answered with a sure $\perp$ to "is x outside both?" answered with a sure $\top$.

Mixed-axis intersection leaves the line

The complement enters destructively the moment it sits inside an intersection. Let $\mathsf{A}^{\pm}$ be FNR-zero at $(0.01, 0)$ and $\mathsf{B}^{\pm}$ FNR-zero at $(0.02, 0)$, and form the difference $\mathsf{A}^{\pm} \setminus \mathsf{B}^{\pm} = \mathsf{A}^{\pm} \cap \complement\mathsf{B}^{\pm}$. The complement $\complement\mathsf{B}^{\pm}$ sits at $(0, 0.02)$ on the FPR-zero axis, so the intersection pairs one operand from each axis. By the corrected intersection formula (6.5), the false-positive rate is *not* the bare product: with $\mathsf{A}^{\pm} = (0.01, 0)$ and $\complement\mathsf{B}^{\pm} = (0, 0.02)$, the outside-both region survives,

$$\varepsilon_{\cap} \approx \varepsilon_A(1-\omega_{\complement B}) = 0.01\times0.98 = 0.0098, \qquad \omega_{\cap} = 1-(1-0)(1-0.02) = 0.02.$$

The composition lands at $\approx (0.0098, 0.02)$, in the two-sided interior, not on either operand's axis. This is the same two-sidedness §6.2 found for the difference directly, as it must be, since the two are the same set.

Proposition 6.3.2 (Mixed-axis intersection escapes the line)**.** *Let* $\mathsf{A}^{\pm}$ *be FNR-zero (*$\varepsilon_A > 0$*,* $\omega_A = 0$*) and* $\mathsf{B}^{\pm}$ *FPR-zero (*$\varepsilon_B = 0$*,* $\omega_B > 0$*). Then* $\mathsf{A}^{\pm} \cap \mathsf{B}^{\pm}$ *is two-sided: by* (6.5) *its FPR is* $w_2\varepsilon_A(1-\omega_B) > 0$*, contributed by the region outside both sets where* $\mathsf{A}^{\pm}$ *false-positives and* $\mathsf{B}^{\pm}$ *accepts, and by* (6.4) *its FNR is* $\omega_B > 0$*. The line is not preserved.*

Proof. Substitute $(\varepsilon_A, 0)$ and $(0, \omega_B)$ into Proposition 6.2.1. The FNR is $1-(1-0)(1-\omega_B) = \omega_B$. In the FPR sum (6.5) the first and third terms carry a factor $\varepsilon_B = 0$ and vanish, leaving $w_2\varepsilon_A(1-\omega_B)$, strictly positive whenever the outside-both region has positive weight. Both output rates are positive, so the result is off both axes. □

The older intuition that this intersection "lands on the FPR-zero axis" is an artifact of approximating $\varepsilon_\cap$ by the product $\varepsilon_A\varepsilon_B = 0$; the linear term is what makes the escape immediate, with no need for a leak in either operand's one-sidedness.

The three regimes

The chapter-6 operations fall into three classes by their action on the perfect filter line.

Axis-preserving. Union and intersection of two FNR-zero sets stay FNR-zero: $\omega = 0$ is preserved (Equation (6.2), Equation (6.4) vanish), even though the surviving FPR is the full three-region rate of §6.2. The FPR-zero side is symmetric.

Axis-swapping. Complement of a one-sided set is one-sided on the other axis (Proposition 6.3.1): the result leaves each axis individually but stays on the line as a whole.

Axis-leaving. Mixed-axis intersection is two-sided (Proposition 6.3.2). Difference $\mathsf{A}^{\pm} \setminus \mathsf{B}^{\pm} = \mathsf{A}^{\pm} \cap \complement\mathsf{B}^{\pm}$ is two-sided exactly when $\mathsf{A}^{\pm}$ and $\mathsf{B}^{\pm}$ lie on the *same* axis (so the complement sends $\mathsf{B}^{\pm}$ to the opposite axis and the intersection is mixed); it stays one-sided only when the operands lie on opposite axes to begin with. These are the compositions that exit the line into the interior, where both directions of report are uncertain.

Why it matters

Whether a composed set sits on the line governs the three later chapters that take the line as their subject. Chapter 7 simplifies for a one-sided structure (an FNR-zero set has NPV = 1 and one nonzero rate to estimate) and exercises the full machinery for a two-sided one. Chapter 14 proves the

entropy floor $-\log_2 \varepsilon$ for FNR-zero sets and achieves it with the Bernoulli Hash Function; a chain that leaves the axis falls outside the regime the bound measures. Chapter 15 builds on the FPR-zero axis, and a mixed-axis intersection with a Bloom filter destroys its certain $\top$. Composition discipline, keeping a chain on one axis, is the price of staying near the floor; Section 6.4 states the discipline as a closure result.

6.4 The Perfect Filter Line

Section 5.4 named the perfect filter line as the union of the two coordinate axes of the rate plane. The composition formulas of this chapter, with the cancellation classification of Section 6.3, let us read the line as an algebraic object: a distinguished subset of $[0,1]^2$ that some operations preserve, some map to itself, and some exit.

Definition 6.4.1 (The perfect filter line)**.** *The* perfect filter line *is* $\mathcal{L} = \mathcal{L}_\omega \cup \mathcal{L}_\varepsilon$*, where the* FNR-zero axis *is* $\mathcal{L}_\omega = \{(\varepsilon, 0) : 0 < \varepsilon \leq 1\}$ *and the* FPR-zero axis *is* $\mathcal{L}_\varepsilon = \{(0, \omega) : 0 < \omega \leq 1\}$*. An approximate set* lies on the line *when its rate point* (ε, ω) *belongs to* $\mathcal{L}$*.*

The axes meet only at the origin $(0,0)$, the exact set, excluded by the strict inequalities: a set on the line achieves certainty in exactly one direction of report, and the origin achieves it in both at the cost of the space approximate sets exist to save. We draw ε horizontal and ω vertical, so $\mathcal{L}_\omega$ is the horizontal arm and $\mathcal{L}_\varepsilon$ the vertical one; Chapter 14 builds on the horizontal arm and Chapter 15 on the vertical.

Proposition 6.4.1 (FNR-zero axis closure)**.** *Let* $\mathsf{A}^{\pm}, \mathsf{B}^{\pm}$ *be FNR-zero, so* $\omega_A = \omega_B = 0$*. Then:*

1. $\mathsf{A}^{\pm} \cup \mathsf{B}^{\pm}$ *is FNR-zero, with* $\varepsilon_\cup = 1 - (1 - \varepsilon_A)(1 - \varepsilon_B)$*;*
2. $\mathsf{A}^{\pm} \cap \mathsf{B}^{\pm}$ *is FNR-zero, with* $\varepsilon_\cap$ *the three-region rate* (6.5)*, equal to* $\varepsilon_A \varepsilon_B$ *when* $A = B$ *or in the large-universe limit;*
3. $\complement\mathsf{A}^{\pm}$ *is FPR-zero, at* $(0, \varepsilon_A)$*;*
4. $\mathsf{A}^{\pm} \setminus \mathsf{B}^{\pm}$ *is* two-sided*, with* $\omega_\setminus = \varepsilon_B$ *and* $\varepsilon_\setminus = w_2 \varepsilon_A (1 - \varepsilon_B) > 0$*.*

So $\mathcal{L}_\omega$ *is closed under union and intersection, mapped to* $\mathcal{L}_\varepsilon$ *by complement, and* not *closed under difference.*

Proof. Parts (1) and (2) set $\omega_A = \omega_B = 0$ in Proposition 6.1.1 and Proposition 6.2.1: the union FNR (6.2) and intersection FNR (6.4) both vanish, while the FPR formulas are unaffected by the operand FNRs. Part (3) is Proposition 6.2.2 at $\omega_A = 0$. For part (4), the difference is $\mathsf{A}^{\pm} \cap \complement\mathsf{B}^{\pm}$; with $\mathsf{B}^{\pm}$ FNR-zero its complement is FPR-zero at $(0, \varepsilon_B)$, so the intersection is mixed-axis and Proposition 6.3.2 applies, giving $\omega_{\setminus} = \varepsilon_B$ and a strictly positive $\varepsilon_{\setminus} = w_2 \varepsilon_A (1 - \varepsilon_B)$ on the region outside both sets. □

Part (4) is the difference's signature: two FNR-zero operands compose to a two-sided set, because the intermediate complement sends $\mathsf{B}^{\pm}$ to the opposite axis and the intersection then mixes axes. The difference stays one-sided only across axes, when $\mathsf{A}^{\pm}$ and $\mathsf{B}^{\pm}$ start on *opposite* arms so the complement aligns them; same-axis operands always produce a two-sided difference.

Proposition 6.4.2 (FPR-zero axis closure)**.** *Dually, if* $\mathsf{A}^{\pm}, \mathsf{B}^{\pm}$ *are FPR-zero then* $\mathsf{A}^{\pm} \cup \mathsf{B}^{\pm}$ *and* $\mathsf{A}^{\pm} \cap \mathsf{B}^{\pm}$ *are FPR-zero,* $\complement\mathsf{A}^{\pm}$ *is FNR-zero, and* $\mathsf{A}^{\pm} \setminus \mathsf{B}^{\pm}$ *is two-sided.*

Proof. By complement-swap from Proposition 6.4.1: an FPR-zero set is the complement of an FNR-zero set, and the rate-plane algebra is symmetric under coordinate exchange, so each conclusion transposes. □

The shape of the line. $\mathcal{L}$ is non-convex: the midpoint of a segment from $(\varepsilon, 0)$ to $(0, \omega)$ is $(\varepsilon/2, \omega/2)$, interior to the square and on neither axis. The two arms meet only at the excised origin, so $\mathcal{L}$ is a wedge of two open segments rather than a connected curve, and no convex mixing of one operand from each arm preserves it, the geometric content of Proposition 6.3.2. This is the precise sense in which the line is a *subalgebra* of the rate plane: closed under same-axis union and intersection, swapped by complement, and exited by mixed-axis intersection and same-axis difference. Same-axis chains compose freely; cross-axis chains leave the line, and the space guarantees of Chapter 14 and Chapter 15 no longer apply to the result.

Remark (The two-pass skeleton)**.** *A natural design pairs a cheap FNR-zero filter* $\mathsf{A}^{\pm}_{-}$ *at* $(\varepsilon_-, 0)$ *with a precise FPR-zero perfect filter* $\mathsf{A}^{\pm}_{+}$ *at* $(0, \omega_+)$ *for the same A, accepting only when both accept. By Proposition 6.3.2 this predicate-intersection is mixed-axis and hence two-sided, with FNR* ω_+ *and a residual FPR* $w_2 \varepsilon_- (1 - \omega_+)$ *on inputs outside the perfect filter's certainty region. The cheap filter discards most non-members and the precise filter certifies the rest, which is the algorithmic skeleton of the perfect hash filter;*

Chapter 15 is what turns the skeleton into a genuinely one-sided FPR-zero construction rather than a two-sided approximation. △

Chapter 14 builds the Bernoulli Hash Function on $\mathcal{L}_\omega$ at the entropy floor, closing the standard Bloom filter's $\log_2 e \approx 1.44$ overhead, and by Proposition 6.4.1 its unions and intersections stay on the axis at predictable rates. Chapter 15 builds the perfect hash filter on $\mathcal{L}_\varepsilon$; mixed-axis composition with a Bloom filter leaves the line and forfeits the FPR-zero certainty, so the designer must respect the closure constraints to keep either chapter's guarantees.

6.5 What Comes Next

The chapter fixed the algebra of set operations: closed-form rates for union, intersection, complement, and difference (Section 6.1, Section 6.2), the classification of which compositions leave the perfect filter line (Section 6.3), and the line as a subalgebra of the rate plane (Section 6.4). Two questions remain for the rest of Part II: how good an approximate set is against a workload, and how small it can be made. Chapter 7 turns the per-block rates into the workload-dependent measures practitioners report, positive and negative predictive value, F_1, recall and precision, feeding the composition rates directly into them (a union of FNR-zero filters is FNR-zero by Proposition 6.4.1, so its PPV follows with no extra work), and it retires the interval-estimator deferral of Section 1.4 with the Wilson, Clopper-Pearson, and Jeffreys constructions. Chapter 8 gives the rates a space-cost analog, the entropy floor $-\log_2 \varepsilon$ bits per element for an FNR-zero set, against which the closure results bound what a composition can cost; Chapter 14 then achieves the floor with the Bernoulli Hash Function, closing the standard Bloom filter's 44% overhead.

Bibliographic Notes

The composition framework. The rate-propagation formulas of Sections 6.1 and 6.2 are the set-level instances of the framework of Towell [Tow26g], which proves the propagation laws for the full algebra, union, intersection, complement, difference, and symmetric difference, under independence of the constituent Bernoulli[bool] outputs, and catalogues which compound operations preserve one-sidedness. That manuscript carries the partition-weighted forms used here for the off-axis rates; this chapter de-

rives each rule from the predicate algebra of Section 5.3 and the gate rules of Section 3.2 and connects it to its Boolean-algebra ancestor.

Bloom filter union and intersection. That two Bloom filters of equal length and shared hash family admit a bitwise OR and AND of their arrays is folklore, but what the result *computes*, and at what rate, is treated in Mitzenmacher and Upfal [MU17, Ch. 5] and surveyed in Broder and Mitzenmacher [BM04]. The union case is clean: the bitwise OR is exactly a filter for $A \cup B$ with $\omega = 0$ preserved. The intersection case is subtle: the bitwise AND does not in general produce a filter for $A \cap B$, admitting ghost members absent from both inputs. Section 6.2 sidesteps the trap by treating intersection at the predicate level, which gives the correct rate without committing to a representation.

De Morgan under noise. The complement-swap of Proposition 6.2.2, that complementing an approximate set exchanges its two rates, is the set-level shadow of the NOT rule of Section 3.2. Classical De Morgan, $\complement(A \cup B) = \complement A \cap \complement B$, still holds at the set level, but the two sides traverse the rate plane by different paths: one touches the FNR-zero axis when both inputs are FNR-zero, the other the FPR-zero axis after the swap. "Different paths, same destination" is what makes the perfect-filter-line classification of Section 6.4 informative about which compound operations stay one-sided and which are forced into the interior.

Perfect-filter lineage. The constructive side of building a set whose rates sit exactly on the FPR-zero axis has roots in the perfect-hashing literature opened by Carter and Wegman [CW79] and is taken up in Chapter 15, which builds on Towell [Tow26m] for the space-optimal construction. The Bloom filter of Bloom [Blo70] occupies the opposite axis and has been the canonical one-sided approximate set since 1970; its behavior under union and intersection is what this chapter's closure analysis formalizes. Chapter 6 sits between the predicate algebra of chapter 5 and the constructive machinery of chapter 15: the algebra of operations, divorced from any representation.

Chapter 7

Classification Measures

7.1 The Setup Revisited

Section 1.4 posed the estimation problem and stopped short of the theory. An approximate set $\mathsf{A}^{\pm} : U \to \text{Bernoulli[bool]}$ arrives with two unknown rates: ε, the chance it reports $\top$ on a $\bot$-input, and ω, the chance it reports $\bot$ on a $\top$-input. Query it on n labelled inputs, split them into the n_0 trials whose truth is $\bot$ and the $n_1 = n - n_0$ whose truth is $\top$, count false positives and false negatives, and report $\hat{\varepsilon} = \text{FP}/n_0$, $\hat{\omega} = \text{FN}/n_1$. Chapter 1 stated the binomial form, the unbiasedness, and a Wald confidence interval, all without proof, and deferred four things: the estimator's exact properties (bias, variance, the MLE identity, efficiency); proper confidence intervals to replace the Wald form; the cause of the sawtooth under-coverage its notebook displayed; and the derived measures, precision and the predictive values, that depend on the workload as the rates do not. This chapter supplies all four, in that order, and closes with a deployed Bloom filter run through the whole toolkit.

Notation. The rates of Section 1.2 carry their usual symbols: ε and ω for the two error rates, $\tau = 1 - \omega$ for the true-positive rate (*recall* or *sensitivity*), and $\eta = 1 - \varepsilon$ for the true-negative rate (*specificity*); the hatted forms are their empirical estimators. Sample sizes are n_0 (truth $\bot$) and n_1 (truth $\top$), so $\hat{\varepsilon} = \text{FP}/n_0$ and $\hat{\omega} = \text{FN}/n_1$, and $z_{\alpha/2}$ is the upper-$\alpha/2$ standard-Normal quantile. The workload prevalence $\pi = \text{P}(X = \top)$ is the fraction of $\top$-inputs in the inference sample; PPV, NPV, and F_1 depend on it and ε, ω do not.

7.2 The Binomial Estimator: Properties

The estimator $\hat{\varepsilon} = \mathrm{FP}/n_0$ that Section 1.4 reported without proof is unbiased for ε, has variance $\varepsilon(1-\varepsilon)/n_0$, equals the maximum-likelihood estimator, and attains the Cramér–Rao lower bound. Five short propositions settle these claims. Of the n trials, the n_0 with truth $\perp$ are the only ones a false positive can come from; for each, let $X_i = 1$ if the noisy bit reports $\top$. Axiom 1 makes the X_i i.i.d. Bernoulli(ε), so $\mathrm{FP} = \sum_{i=1}^{n_0} X_i \sim \mathrm{Binomial}(n_0, \varepsilon)$ and $\hat{\varepsilon} = \mathrm{FP}/n_0$. The arguments for $\hat{\omega} = \mathrm{FN}/n_1$ are identical under $\varepsilon \to \omega$, $n_0 \to n_1$.

Proposition 7.2.1 (Unbiasedness)**.** $\mathrm{E}[\hat{\varepsilon}] = \varepsilon$.

Proof. By linearity, $\mathrm{E}[\mathrm{FP}] = \sum_{i=1}^{n_0} \mathrm{E}[X_i] = n_0\varepsilon$, and dividing by the constant n_0 gives $\mathrm{E}[\hat{\varepsilon}] = \varepsilon$. □

The bias is zero at every ε and every $n_0 \geq 1$, not merely in the limit.

Proposition 7.2.2 (Variance)**.** $\mathrm{Var}(\hat{\varepsilon}) = \varepsilon(1-\varepsilon)/n_0$.

Proof. Each X_i has $\mathrm{Var}(X_i) = \varepsilon - \varepsilon^2 = \varepsilon(1-\varepsilon)$; independence makes $\mathrm{Var}(\mathrm{FP}) = n_0\varepsilon(1-\varepsilon)$, and the scaling rule $\mathrm{Var}(\mathrm{FP}/n_0) = \mathrm{Var}(\mathrm{FP})/n_0^2$ gives the result. □

The standard error shrinks as $1/\sqrt{n_0}$, and the numerator $\varepsilon(1-\varepsilon)$ peaks at $\varepsilon = 1/2$ and vanishes at the boundaries: the rate is hardest to pin down when the classifier is maximally noisy, the fact that returns in Section 7.3 as the reason Wald intervals deform near 0 and 1.

Proposition 7.2.3 (The MLE is the empirical estimator)**.** *The maximum-likelihood estimator of ε from* $\mathrm{FP} \sim \mathrm{Binomial}(n_0, \varepsilon)$ *is* $\hat{\varepsilon}_{\mathrm{MLE}} = \mathrm{FP}/n_0 = \hat{\varepsilon}$.

Proof. The log-likelihood $\log L(\varepsilon) = \mathrm{const} + \mathrm{FP}\log\varepsilon + (n_0 - \mathrm{FP})\log(1-\varepsilon)$ has derivative $\mathrm{FP}/\varepsilon - (n_0-\mathrm{FP})/(1-\varepsilon)$, whose zero solves $\mathrm{FP}(1-\varepsilon) = (n_0-\mathrm{FP})\varepsilon$, i.e. $\varepsilon = \mathrm{FP}/n_0$. The second derivative $-\mathrm{FP}/\varepsilon^2 - (n_0-\mathrm{FP})/(1-\varepsilon)^2$ is negative on $(0,1)$, so the critical point is the unique interior maximum; the boundary counts $\mathrm{FP} \in \{0, n_0\}$ give $\hat{\varepsilon}_{\mathrm{MLE}} \in \{0, 1\}$, again FP/n_0. □

The empirical estimator is what likelihood theory asks for, not merely a convenient choice. It also equals the first-moment method-of-moments estimator (equate $\mathrm{E}[\mathrm{FP}] = n_0\varepsilon$ to the observed count). The coincidence is special to one-parameter exponential families like the binomial, whose variance is fixed by the mean; in a two-parameter model such as Beta(a, b)

the first moment is one equation in two unknowns, the moment hierarchy must reach the second moment to identify both parameters, and maximum likelihood and the method of moments generally diverge.

Proposition 7.2.4 (Fisher information and the Cramér–Rao bound)**.** *The Fisher information is* $I(\varepsilon) = n_0/[\varepsilon(1-\varepsilon)]$*, so every unbiased estimator* T *of* ε *satisfies* $\mathrm{Var}(T) \geq 1/I(\varepsilon) = \varepsilon(1-\varepsilon)/n_0$*.*

Proof. The second derivative from Proposition 7.2.3, in expectation under $\mathrm{FP} \sim \mathrm{Binomial}(n_0, \varepsilon)$ with $\mathrm{E}[\mathrm{FP}] = n_0\varepsilon$, gives

$$I(\varepsilon) = -\mathrm{E}\left[\frac{\partial^2 \log L}{\partial \varepsilon^2}\right] = \frac{n_0}{\varepsilon} + \frac{n_0}{1-\varepsilon} = \frac{n_0}{\varepsilon(1-\varepsilon)}.$$

The binomial family is regular (fixed support, twice-differentiable log-likelihood on $(0,1)$), so the Cramér–Rao inequality applies and $\mathrm{Var}(T) \geq 1/I(\varepsilon)$. □

Proposition 7.2.5 (Efficiency)**.** $\hat{\varepsilon} = \mathrm{FP}/n_0$ *attains the Cramér–Rao bound and is an efficient unbiased estimator of* ε*.*

Proof. Proposition 7.2.2 gives $\mathrm{Var}(\hat{\varepsilon}) = \varepsilon(1-\varepsilon)/n_0$, which equals the bound $1/I(\varepsilon)$ of Proposition 7.2.4. □

No unbiased estimator of ε does better in mean-squared error within the binomial model: a candidate reporting a lower variance is biased, trading bias for variance. What remains is to turn the point estimate into a confidence statement that holds at every n_0, not only asymptotically, which Section 7.3 takes up.

7.3 Confidence Intervals Properly

Section 7.2 fixed $\hat{\varepsilon} = \mathrm{FP}/n_0$ as the unbiased, efficient MLE. A point estimate is not a confidence statement, and the Wald interval of Section 1.4 fails in exactly the regime a practitioner most needs it. This section diagnoses the failure and derives three alternatives.

Definition 7.3.1 (Wald confidence interval)**.** *At level* $1-\alpha$*, the Wald interval for* ε *is* $\hat{\varepsilon} \pm z_{\alpha/2}\sqrt{\hat{\varepsilon}(1-\hat{\varepsilon})/n_0}$*.*

It reads the central-limit Normal $\hat{\varepsilon} \sim \mathrm{Normal}(\varepsilon, \varepsilon(1-\varepsilon)/n_0)$ off the sampling distribution and plugs $\hat{\varepsilon}$ in for ε in the variance. Correct as $n_0 \to \infty$, it fails at finite n_0 near a boundary for two reasons. *Boundary:* the

symmetric radius can push the lower endpoint below 0 or the upper above 1, assigning probability to impossible rates and needing ad hoc clipping. At $\hat{\varepsilon} = 0$ it collapses to the degenerate point $[0, 0]$. *Skew:* Binomial(n_0, ε) is symmetric only at $\varepsilon = 1/2$; near a boundary the central limit theorem has not overcome the binomial skew, and the symmetric interval miscalibrates, dropping below nominal coverage (near 0.88 at $n_0 = 100$, $\varepsilon = 0.05$ rather than 0.95). A third, finer effect is the *sawtooth*: coverage as a function of n_0 oscillates jaggedly because the interval is continuous while FP is integer-valued, so the cover/miss boundary shifts in discrete jumps. The sawtooth afflicts every interval on a discrete distribution; Wald is unusual only in how far its dips fall below nominal.

Wilson, Clopper–Pearson, Jeffreys

Wilson inverts the *score* statistic, which keeps the true ε in the variance rather than its estimate. The confidence set $\{\varepsilon : |Z(\varepsilon)| \le z\}$ with $Z(\varepsilon) = (\hat{\varepsilon} - \varepsilon)/\sqrt{\varepsilon(1-\varepsilon)/n_0}$ and $z = z_{\alpha/2}$ squares to the quadratic

$$\varepsilon^2\Big(1 + \tfrac{z^2}{n_0}\Big) - 2\varepsilon\Big(\hat{\varepsilon} + \tfrac{z^2}{2n_0}\Big) + \hat{\varepsilon}^2 \le 0,$$

whose two roots are the interval endpoints.

Definition 7.3.2 (Wilson confidence interval)**.** *At level* $1-\alpha$ *with* $z = z_{\alpha/2}$*,*

$$\frac{\hat{\varepsilon} + z^2/(2n_0)}{1 + z^2/n_0} \;\pm\; \frac{z}{1 + z^2/n_0}\sqrt{\frac{\hat{\varepsilon}(1-\hat{\varepsilon})}{n_0} + \frac{z^2}{4n_0^2}}.$$

The center is $\hat{\varepsilon}$ shrunk toward $1/2$, and the radius carries a correction $z^2/(4n_0^2)$ that keeps the interval non-degenerate at $\hat{\varepsilon} \in \{0, 1\}$: at $\hat{\varepsilon} = 0$ it gives $[0, z^2/(n_0 + z^2)]$, admitting small positive rates. Both endpoints lie in $[0, 1]$ by construction, killing the boundary problem, and the shift toward $1/2$ corrects the direction the symmetric Wald interval got wrong.

Clopper–Pearson abandons the Normal approximation and inverts the exact binomial test: ε is included iff the observed FP is consistent with Binomial(n_0, ε) at level α. The endpoints are Beta quantiles, via the binomial–Beta identity $\mathrm{P}\left(\text{Binomial}(n_0, \varepsilon) \ge k\right) = \mathrm{P}\left(\text{Beta}(k, n_0 - k + 1) \le \varepsilon\right)$.

Definition 7.3.3 (Clopper–Pearson confidence interval)**.** *At level* $1-\alpha$*, the endpoints are* $L = F^{-1}_{\text{Beta}(\text{FP},\, n_0 - \text{FP} + 1)}(\alpha/2)$ *and* $U = F^{-1}_{\text{Beta}(\text{FP}+1,\, n_0 - \text{FP})}(1 - \alpha/2)$*, with* $L = 0$ *if* $\text{FP} = 0$ *and* $U = 1$ *if* $\text{FP} = n_0$*.*

Called *exact* for using the exact binomial CDF, it guarantees coverage at least $1-\alpha$ but, because the binomial is discrete, over-covers at most ε, sometimes substantially. Conservatism is the price of the guarantee.

Jeffreys reports a posterior credible interval under the prior $\pi_J(\varepsilon) \propto \sqrt{I(\varepsilon)} \propto \varepsilon^{-1/2}(1-\varepsilon)^{-1/2}$, the Beta$(1/2, 1/2)$ kernel; by conjugacy the posterior is Beta$(\mathrm{FP}+1/2, n_0-\mathrm{FP}+1/2)$.

Definition 7.3.4 (Jeffreys credible interval)**.** *The central* $1-\alpha$ *quantile interval of* Beta$(\mathrm{FP}+1/2,\ n_0-\mathrm{FP}+1/2)$.

It resembles Clopper–Pearson with a half-integer shift but treats the endpoints symmetrically and is non-degenerate at the boundaries; its frequentist coverage is close to nominal at small n_0 without Clopper–Pearson's heavy conservatism.

Remark (The log-odds scale)**.** *A fourth boundary-robust route transforms before forming a symmetric interval. The logit* $g(\varepsilon) = \log[\varepsilon/(1-\varepsilon)]$ *has* $g'(\varepsilon) = 1/[\varepsilon(1-\varepsilon)]$*, so by the delta method* $\mathrm{Var}[g(\hat{\varepsilon})] \approx 1/[n_0\,\varepsilon(1-\varepsilon)]$*. The transform sends the boundaries to* $\pm\infty$*, where the raw estimator's variance collapses to zero and the Wald interval degenerates; a symmetric interval built on the logit scale and mapped back by the inverse logit stays strictly inside* $(0,1)$ *and is asymmetric in* ε*. This is the re-parametrization behind logistic-regression intervals.* △

Comparison and recommendation

	Wald	Wilson	Clopper–Pearson	Jeffreys
Coverage, $n_0 = 100$, $\varepsilon = 0.05$	≈ 0.88	≈ 0.97	≥ 0.95	≈ 0.94
Coverage, $n_0 = 1000$, $\varepsilon = 0.05$	≈ 0.95	≈ 0.95	≥ 0.95	≈ 0.95
Interval at $\hat{\varepsilon} = 0$	$[0, 0]$	$[0,\ z^2/(n_0+z^2)]$	$[0,\ U_{\mathrm{CP}}]$	$[L_J,\ U_J]$
Endpoints in $[0,1]$	no	yes	yes	yes
Approximation	Normal	Normal (score)	exact binomial	Bayesian

Table 7.1: Four binomial confidence intervals at nominal $1-\alpha = 0.95$. The $n_0 = 100$ figures are exact single-point binomial coverages at the stated (n_0, ε); at $n_0 = 1000$ all four are near nominal. Coverage oscillates along a sawtooth in n_0 (Brown, Cai, and DasGupta 2001), so neighbouring sizes differ. Wald is the only interval whose dips fall far enough below nominal to mislead, and the only one that degenerates at the boundary.

The Wilson interval is this book's default and is what Section 7.4 feeds into the delta method: it corrects the Wald boundary failure without Clopper–Pearson's conservatism and needs only a Normal-quantile table. Jeffreys is the Bayesian alternative with excellent frequentist coverage; Clopper–Pearson is the conservative choice when an absolute coverage floor is required; Wald is acceptable only at large n_0 away from the boundaries, where all four agree. Section 7.4 now takes up the derived measures and the delta-method translation of these intervals.

7.4 Derived Measures

The fourth deferred item is the derived measures. Precision, recall, F_1, and the two predictive values appear throughout binary classification and information retrieval, and Section 1.2 noted that precision depends on a quantity the noisy bit does not control: the workload's prevalence of $\top$. This section makes that precise, writing each measure as a function of $(\varepsilon, \omega, \pi)$ and propagating the binomial confidence intervals through the delta method.

Definitions

Apply a noisy bit with rates ε, ω to a workload of n inputs whose truth is i.i.d. Bernoulli(π) with $\pi = \mathrm{P}(X = \top)$. The expected contingency-table counts are

$$\mathrm{E}[\mathrm{TP}] = n\pi(1-\omega), \quad \mathrm{E}[\mathrm{FN}] = n\pi\omega, \quad \mathrm{E}[\mathrm{FP}] = n(1-\pi)\varepsilon, \quad \mathrm{E}[\mathrm{TN}] = n(1-\pi)(1-\varepsilon),$$

the first row summing to $n\pi$, the second to $n(1-\pi)$. Recall is the row-conditional rate recall $= \tau = 1 - \omega$, with π cancelling: recall is a property of the noisy bit alone. The predictive values and F_1 are column quantities, and π does not cancel.

Definition 7.4.1 (Positive predictive value)**.** *At workload prevalence π,*

$$\mathrm{PPV} = \text{precision} = \frac{\pi(1-\omega)}{\pi(1-\omega) + (1-\pi)\varepsilon},$$

the ratio of $\mathrm{E}[\mathrm{TP}]$ *to the $\top$-output column total* $\mathrm{E}[\mathrm{TP} + \mathrm{FP}]$.

Definition 7.4.2 (Negative predictive value)**.** *At workload prevalence π,*

$$\mathrm{NPV} = \frac{(1-\pi)(1-\varepsilon)}{(1-\pi)(1-\varepsilon) + \pi\omega},$$

the mirror of PPV on the $\perp$-output column.

Definition 7.4.3 (F_1-score)**.** *F_1 is the harmonic mean of precision and recall,*

$$F_1 = \frac{2\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\pi(1-\omega)}{\pi(1-\omega) + \pi + (1-\pi)\varepsilon},$$

the second form using $\pi(1-\omega) + \pi\omega = \pi$.

F_1 weights precision and recall equally; the weighted family $F_\beta = (1 + \beta^2)\text{precision}\text{recall}/(\beta^2\text{precision} + \text{recall})$ tilts toward recall for $\beta > 1$ and precision for $\beta < 1$, recovering F_1 at $\beta = 1$. Replacing the parameters by their estimates turns each formula into a point estimator.

Prevalence dependence

Read PPV as a function of π at fixed rates. At $\pi = 0$ the numerator vanishes, so $\text{PPV}(0) = 0$ when $\varepsilon > 0$: a workload with no genuine positives reports only false positives. At $\pi = 1$ the denominator collapses to the numerator and $\text{PPV}(1) = 1$. Between, PPV rises monotonically, steeply near $\pi = 0$ when ε is appreciable; NPV runs the mirror story, $\text{NPV}(0) = 1$, $\text{NPV}(1) = 0$. The framing is the point. The pair (ε, ω) is intrinsic to the classifier; precision is a joint property of classifier and workload, and a vendor who reports only precision is reporting what their evaluation workload happened to look like. The book uses (ε, ω) as the primary description for exactly this reason.

Estimation and the delta method

The estimator $\widehat{\text{PPV}} = \text{TP}/(\text{TP}+\text{FP})$ is a ratio of two independent binomial counts: conditioning on the row totals, $\text{TP} \sim \text{Binomial}(n_\top, \tau)$ and $\text{FP} \sim \text{Binomial}(n_\bot, \varepsilon)$, independent by Axiom 1. A ratio's expectation is not the ratio of expectations, so $\widehat{\text{PPV}}$ is biased (small when both means are large, and consistent as $n \to \infty$), and its variance is read off by the delta method rather than directly.

Theorem 7.4.1 (Delta method)**.** *If $\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} \text{Normal}(0, \sigma^2(\theta))$ and g is differentiable at θ with $g'(\theta) \neq 0$, then $\sqrt{n}(g(\hat{\theta}) - g(\theta)) \xrightarrow{d} \text{Normal}(0, [g'(\theta)]^2\sigma^2(\theta))$.*

This is the first-order Taylor expansion $g(\hat{\theta}) \approx g(\theta) + g'(\theta)(\hat{\theta} - \theta)$ with Slutsky's lemma; a proof is in Casella and Berger [CB02, Ch. 5]. The multivariate form replaces $g'(\theta)$ by the gradient and σ^2 by a covariance matrix, giving the quadratic form $\nabla g^\top \Sigma \nabla g$.

Proposition 7.4.1 (Delta-method variance of $\widehat{\mathrm{PPV}}$)**.** *With* $\bar{t} = n_\top \tau$, $\bar{f} = n_\perp \varepsilon$, $\sigma_t^2 = n_\top \tau(1-\tau)$, $\sigma_f^2 = n_\perp \varepsilon(1-\varepsilon)$,

$$\operatorname{Var}(\widehat{\mathrm{PPV}}) \approx \frac{\bar{f}^{\,2}\sigma_t^2 + \bar{t}^{\,2}\sigma_f^2}{(\bar{t}+\bar{f})^4}.$$

Proof. For $g(t,f) = t/(t+f)$, the gradient is $\nabla g = (t+f)^{-2}(f,-t)$. The covariance of $(\mathrm{TP},\mathrm{FP})$ is $\operatorname{diag}(\sigma_t^2,\sigma_f^2)$ by independence, and the quadratic form $\nabla g^\top \operatorname{diag}(\sigma_t^2,\sigma_f^2)\nabla g$ at the mean $(\bar{t},\bar{f})$ gives the stated formula. □

A Wald-style PPV interval is $\widehat{\mathrm{PPV}} \pm z_{\alpha/2}\sqrt{\widehat{\operatorname{Var}}(\widehat{\mathrm{PPV}})}$, which inherits the boundary and skew failures of the Wald rate interval; the remedy is a Wilson-style score interval on the column-conditional binomial $\mathrm{TP} \mid \mathrm{TP}+\mathrm{FP} \sim \mathrm{Binomial}(\mathrm{TP}+\mathrm{FP}, \mathrm{PPV})$, which stays inside $[0,1]$ by construction, or a bootstrap over the contingency table.

F_1 via the delta method

Writing $F_1 = 2\mathrm{TP}/(\mathrm{TP}+\mathrm{FP}+n_\top)$ (using $\mathrm{FN} = n_\top - \mathrm{TP}$) and applying Theorem 7.4.1 to $h(t,f) = 2t/(t+f+n_\top)$ gives

$$\operatorname{Var}(F_1) \approx \frac{4(\bar{f}+n_\top)^2\sigma_t^2 + 4\bar{t}^{\,2}\sigma_f^2}{(\bar{t}+\bar{f}+n_\top)^4},$$

differing from the PPV variance only by the $n_\top$ added to the TP-gradient component.

Example 2 (F_1 with delta-method CI) *Take* $\varepsilon = 0.05$, $\omega = 0.10$, $\pi = 0.1$, $n = 1000$, *so* $n_\top = 100$, $n_\perp = 900$. *Then* recall $= 0.90$ *and*

$$\text{precision} = \frac{0.1\cdot 0.9}{0.1\cdot 0.9 + 0.9\cdot 0.05} = \frac{0.09}{0.135} = \frac{2}{3}, \qquad F_1 = \frac{2\cdot(2/3)\cdot 0.9}{2/3+0.9} \approx 0.766.$$

With $\bar{t} = 90$, $\bar{f} = 45$, $\sigma_t^2 = 100\cdot 0.9\cdot 0.1 = 9$, $\sigma_f^2 = 900\cdot 0.05\cdot 0.95 = 42.75$, *and* $s = 90+45+100 = 235$,

$$\operatorname{Var}(F_1) \approx \frac{4\cdot 145^2\cdot 9 + 4\cdot 90^2\cdot 42.75}{235^4} = \frac{756900 + 1385100}{3{,}049{,}800{,}625} \approx 7.02\times 10^{-4},$$

a standard deviation of 0.0265, *so the* 95% *interval is* $0.766 \pm 1.96\cdot 0.0265 \approx [0.714, 0.818]$. *The interval is narrower than the analogous PPV interval because the* $n_\top$ *in the* F_1 *denominator damps the response to a change in either count.*

Bloom filter PPV under workload

The prevalence dependence is sharp enough to drive a concrete case. Take a Bloom filter (Definition 5.2.1, Section 5.2) with $\omega = 0$ and $\varepsilon = 0.01$, deployed on two workloads. At prevalence $\pi = 0.01$,

$$\mathrm{PPV} = \frac{0.01 \cdot 1}{0.01 \cdot 1 + 0.99 \cdot 0.01} = \frac{0.01}{0.0199} \approx 0.503,$$

so half of all positive reports are false: the filter is right on every $\top$ (by $\omega = 0$) but wrong on 1% of the much larger $\bot$ population, and the two roughly match at the column total. At $\pi = 0.5$,

$$\mathrm{PPV} = \frac{0.5 \cdot 1}{0.5 \cdot 1 + 0.5 \cdot 0.01} = \frac{0.5}{0.505} \approx 0.990.$$

Same filter, same (ε, ω), PPV moving by nearly a factor of two across workloads. "The Bloom filter has precision 0.99" is uninformative without the workload; the (ε, ω) claim is workload-free. Section 7.5 runs the inverse problem, recovering a PPV interval from an observed false-positive count at known prevalence.

7.5 Worked Application: Bloom Filter Diagnostics

One deployment problem exercises the whole toolkit: a spam filter implemented as a Bloom filter, with a question, is the filter precise enough on this workload, and if not, what to change.

Scenario and observation

The filter (Definition 5.2.1, Section 5.2) has parameters $(m, k, n) = (50000, 7, 5000)$, so $m/n = 10$, $k = 7$ put it in the optimal regime of Theorem 5.2.1, and (5.1) at $kn/m = 0.7$ predicts $\varepsilon \approx (1 - e^{-0.7})^7 \approx 0.0082$, with $\omega = 0$ by construction. A workload of $N = 10000$ messages arrives, $n_\top = 1000$ spam and $n_\bot = 9000$ not; the filter reports $\top$ on all spam and on FP $= 92$ of the non-spam, so

$$\hat{\varepsilon} = \frac{92}{9000} \approx 0.01022,$$

about 25% above the design value. The Wilson 95% interval (Definition 7.3.2) at $z = 1.96$ has center $(0.01022 + 1.96^2/18000)/(1 + 1.96^2/9000) \approx 0.01043$

and half-width

$$\frac{1.96}{1.000427}\sqrt{\frac{0.01022 \cdot 0.98978}{9000} + \frac{1.96^2}{4 \cdot 9000^2}} \approx 0.00209,$$

giving $[0.00834, 0.01252]$. The design $\varepsilon = 0.0082$ sits just below the lower endpoint: at $n_\perp = 9000$ the interval is narrow enough to detect the 25% deviation, plausibly from hash collisions that mildly violate the uniform-hash assumption or an item count above the design n. The deployment ε is an empirical quantity to measure, not a design parameter to trust; we carry $\hat{\varepsilon} = 0.0102$ forward.

PPV and its interval

With $\hat{\pi} = n_\top/N = 0.1$, Definition 7.4.1 gives

$$\widehat{\mathrm{PPV}} = \frac{0.1 \cdot 1}{0.1 \cdot 1 + 0.9 \cdot 0.01022} = \frac{0.1}{0.10920} \approx 0.9158,$$

so about 92% of flagged messages are spam. The delta-method variance (Proposition 7.4.1) simplifies because $\omega = 0$ kills the σ_t^2 term: with $\bar{t} = 1000$, $\bar{f} = 92$, $\sigma_f^2 = 9000 \cdot 0.01022 \cdot 0.98978 \approx 91.06$,

$$\mathrm{Var}\,(\widehat{\mathrm{PPV}}) \approx \frac{1000^2 \cdot 91.06}{1092^4} \approx 6.40 \times 10^{-5},$$

a standard error of 0.0080, so the 95% interval is $0.9158 \pm 1.96 \cdot 0.0080 \approx [0.900, 0.932]$. The boundary concern of Section 7.3 does not bite ($\widehat{\mathrm{PPV}}$ is far from 0 and 1, the conditional samples are large). All uncertainty here comes from the false-positive count; were $\omega > 0$, recall would be random and both variance terms would contribute.

The diagnostic

Suppose the requirement is PPV > 0.95. The estimate 0.9158 falls short and the interval $[0.900, 0.932]$ excludes 0.95: a fix is needed, and the interval, not the point, is what establishes it. A point estimate of 0.9158 alone would have suggested the target was nearly met; the interval rules out 0.95 at 95% confidence. Definition 7.4.1 has three knobs, and $\omega = 0$ already, leaving two routes.

Route A, re-tune. Lower ε. Solving $0.95 = 0.1/(0.1 + 0.9\varepsilon)$ gives $\varepsilon \approx 0.00585$; at $\varepsilon = 0.005$ the PPV is $0.1/(0.1 + 0.9 \cdot 0.005) \approx 0.957$, above target. Reaching 0.005 from 0.0102 by (5.1) costs about 16% more bits at

fixed k, a one-time hardware decision that buys an ε surviving the worst-case workload.

Route B, wait. Keep the filter and bound the acceptable workload. Solving $0.95 = \pi/(\pi + (1-\pi)\cdot 0.01022)$ gives $\pi \approx 0.163$; at $\pi = 0.2$ the PPV is ≈ 0.961. Doing nothing structural is acceptable once the spam rate rises above about one in six, but it trades capital cost for the assumption that the workload mix will shift, which the deployment does not control.

The choice is operational, not statistical. Both routes used the same three tools, (5.1) for the design rate, the Wilson interval for its empirical estimate, and Definition 7.4.1 for the translation into precision at prevalence. What the chapter added beyond the chapter-5 design formulas is the honest accounting of uncertainty that turns "nearly met" into "ruled out."

Bibliographic Notes

The classification-measures manuscript. The treatment of PPV, NPV, recall, and F_1 as functions of $(\varepsilon, \omega, \pi)$, with delta-method variances and confidence intervals, follows Towell [Tow26d], the primary source for Section 7.4: the predictive-value expressions, the prevalence-dependence diagnostic, the bivariate computation of Proposition 7.4.1, and the F_1 derivation are all from it. The chapter's addition is to connect that machinery to the Part II approximate-set framework through the predicate $\mathsf{A}^{\pm} : U \to \text{Bernoulli[bool]}$ and to work the deployed Bloom-filter diagnostic of Section 7.5.

The binomial confidence intervals. The interval of Definition 7.3.2 is due to Wilson [Wil27], who inverted the score statistic, keeping ε free in the variance and solving the resulting quadratic, rather than fixing $\varepsilon = \hat{\varepsilon}$ as the Wald form does. It is the standard modern recommendation for the binomial proportion at small n_0 or near a boundary. Definition 7.3.3 is due to Clopper and Pearson [CP34], who introduced test inversion in its cleanest form, taking the interval to be the ε values a level-α binomial test fails to reject; the exact binomial CDF forces coverage strictly above $1-\alpha$ at most rates, the over-coverage that suits safety-critical use. The Jeffreys interval of Definition 7.3.4 predates the Bayesian terminology under which it is now derived, and its higher-order coverage-matching property is a result in asymptotic inference.

The delta method. The asymptotic-Normality result for smooth functions of an asymptotically Normal estimator (Theorem 7.4.1) follows from

the first-order Taylor expansion and Slutsky's lemma; a proof in the form used for the PPV and F_1 derivations is in Casella and Berger [CB02, Ch. 5], with the multivariate version and regularity conditions.

The prevalence question. The prevalence-dependence of PPV and NPV at fixed (ε, ω), the central caveat of Section 7.4, drives the medical diagnostic-testing literature noted in Section 1.5: a test with high sensitivity and specificity can still have low PPV when the condition is rare, because PPV is the classifier-and-workload combination, not the classifier alone. This chapter's contribution is to make the dependence explicit and to give a proper interval for any derived quantity at any specified prevalence.

Chapter 8

Information-Theoretic Bounds

8.1 Shannon Entropy Refresher

The chapter's two results, the BSC capacity (Section 8.2) and the entropy floor for approximate sets (Section 8.3), rest on a small slice of Shannon's information theory: the entropy of a random variable, its binary special case, and mutual information. We collect them and fix notation; Cover and Thomas [CT06] is the depth reference.

Entropy. For a discrete X on a finite alphabet $\mathcal{X}$, the *Shannon entropy* is

$$H(X) \;=\; -\sum_{x\in\mathcal{X}} \mathrm{P}\left(X=x\right)\,\log_2 \mathrm{P}\left(X=x\right), \tag{8.1}$$

with $0\log_2 0 = 0$, measured in bits. Operationally $H(X)$ is the average number of yes/no questions an optimal binary search needs to identify a draw: 0 for a constant, $\log_2|\mathcal{X}|$ for the uniform distribution. By Shannon's source-coding theorem it is also the tight lower bound on the expected length of any prefix-free code for X, the reading Section 8.3 uses.

Binary entropy. For $X \sim \text{Bernoulli}(p)$, (8.1) collapses to

$$H(p) \;=\; -p\log_2 p - (1-p)\log_2(1-p), \qquad p\in[0,1], \tag{8.2}$$

overloading H (a random variable in $H(X)$, a scalar in $H(p)$; context disambiguates). Three properties recur in this chapter and chapter 14: H is

strictly concave ($H''(p) = -1/[p(1-p)\ln 2] < 0$); it peaks at $H(1/2) = 1$ (a fair coin is the most uncertain bit); and $H(0) = H(1) = 0$. It is symmetric, $H(p) = H(1-p)$, with $H(0.01) \approx 0.081$, $H(0.1) \approx 0.469$. Distinct from $H(\varepsilon)$ is the *self-information* (surprisal) $-\log_2 \varepsilon$ of a single event of probability ε: the floor of Section 8.3 is $-\log_2 \varepsilon$ bits per element, the cost of one such event, not the indicator entropy $H(\varepsilon) \approx -\varepsilon \log_2 \varepsilon$ that weights it by ε.

Mutual information and capacity. For jointly distributed X, Y, the *mutual information*

$$I(X;Y) \;=\; H(X) - H(X \mid Y) \;=\; H(Y) - H(Y \mid X) \tag{8.3}$$

measures how much observing one reduces uncertainty about the other, with $H(X \mid Y) = \sum_y \mathrm{P}(Y = y)\, H(X \mid Y = y)$ the conditional entropy. Equivalently

$$I(X;Y) \;=\; \sum_{x,y} \mathrm{P}(X = x, Y = y) \log_2 \frac{\mathrm{P}(X = x, Y = y)}{\mathrm{P}(X = x)\,\mathrm{P}(Y = y)}, \tag{8.4}$$

a Kullback–Leibler divergence between the joint and the product of marginals, hence nonnegative and zero iff X, Y are independent. For a channel with fixed $\mathrm{P}(Y \mid X)$, $I(X;Y)$ depends on the input distribution, and the *capacity* is the maximum,

$$C \;=\; \max_{\mathrm{P}(X)} I(X;Y), \tag{8.5}$$

the most information one channel use can convey. Section 8.2 evaluates it for the BSC of Section 2.1 and gets $C = 1 - H(\varepsilon)$; Section 8.3 applies the source-coding reading to approximate-set representations and gets the $-\log_2 \varepsilon$ floor; Section 8.4 measures the standard Bloom filter's $\log_2 e \approx 1.443$ overhead above it.

8.2 Channel Capacity in the BSC

The binary symmetric channel of Section 2.1 reproduces its Boolean input with probability $1 - \varepsilon$ and flips it with probability ε regardless of the value sent, the channel matrix being $Q = \left(\begin{smallmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{smallmatrix}\right)$ (Definition 2.1.1). Its capacity is the quantitative information-theoretic summary chapter 2 left implicit.

Proposition 8.2.1 (BSC capacity)**.** *The capacity of the BSC with crossover $\varepsilon \in (0,1)$ is*

$$C \;=\; 1 - H(\varepsilon) \;=\; 1 + \varepsilon \log_2 \varepsilon + (1-\varepsilon)\log_2(1-\varepsilon) \quad \textit{bits per use,} \tag{8.6}$$

attained by the uniform input $\mathrm{P}(X = \top) = \mathrm{P}(X = \bot) = 1/2$.

Proof. Compute $I(X;Y) = H(Y) - H(Y \mid X)$ (the second form of (8.3)), with input distribution $\mathrm{P}(X = \top) = q$. Given $X = \top$, the output is Bernoulli(ε) with entropy $H(\varepsilon)$ by (8.2); the same holds for $X = \bot$ since the rows of Q are permutations, so $H(Y \mid X) = qH(\varepsilon) + (1-q)H(\varepsilon) = H(\varepsilon)$, independent of q. The output marginal is $\mathrm{P}(Y = \top) = q(1-\varepsilon) + (1-q)\varepsilon = \varepsilon + q(1-2\varepsilon)$, which equals $1/2$ at $q = 1/2$, where $H(Y) = H(1/2) = 1$ is maximal; for any other q strict concavity gives $H(Y) < 1$. Hence $I(X;Y) = H(\varepsilon + q(1-2\varepsilon)) - H(\varepsilon)$ is maximized at $q = 1/2$, giving $C = 1 - H(\varepsilon)$. □

Once ε is fixed the channel admits at most C bits per use, and by Shannon's noisy-channel coding theorem this ceiling is asymptotically achievable: for any rate $R < C$ there are long block codes with decoding error below any ε. The closed form has a sharp shape. At $\varepsilon = 0$ the channel is noiseless, $H(0) = 0$ and $C = 1$. At $\varepsilon = 1/2$ the output is a fair coin independent of the input, $H(1/2) = 1$ and $C = 0$. At $\varepsilon = 1$ the channel always flips, which a receiver undoes by relabeling $\top \leftrightarrow \bot$, so $H(1) = 0$ and $C = 1$ again; $C(\varepsilon)$ is symmetric about $\varepsilon = 1/2$. The curve is steep near 0 (the first crossover noise costs little: $C(0.01) \approx 0.919$) and shallow near $1/2$ (the last noise costs almost everything: $C(0.1) \approx 0.531$, $C(0.4) \approx 0.029$).

The same source-coding principle drives the rest of the chapter through a different question: how many bits represent an approximate set at target ε? Any prefix-free encoding of a source emitting n symbols of entropy H each needs at least nH bits on average, and Section 8.3 applies this to the bit string an approximate set stores, with the constrained per-element entropy giving a floor of $-\log_2 \varepsilon$ bits per element for any FNR-zero representation. Capacity and the floor are two faces of one principle, with H holding throughout.

8.3 The Entropy Floor for Approximate Sets

The bit-budget comparison of Section 5.4 placed the Bloom filter at roughly 9.6 bits per item at $\varepsilon = 0.01$ against a floor near 6.64. This section turns that contrast into a theorem: any approximate set with $\omega = 0$ and false-positive rate at most ε, in any representation whatever, must store at least $\log_2(1/\varepsilon)$ bits per element on average. The Bloom filter pays a constant factor above this floor; chapter 14 meets it.

Setup

Fix a universe U with $|U| = N$ and a target $A \subseteq U$ with $|A| = n$. An *approximate-set representation* is a pair (R, query) with $R \in \{0,1\}^L$ a bit string and $\mathrm{query}(R, \cdot) : U \to \{\top, \bot\}$ a deterministic membership algorithm; L is the cost to bound. Following Definition 5.4.1 we restrict to the FNR-zero family containing the Bloom filter: $\mathrm{query}(R, x) = \top$ for every $x \in A$ (no inserted element is missed), and at most a ε fraction of the $N - n$ non-members is accepted. The *accepted set* $A_R = \{x : \mathrm{query}(R, x) = \top\}$ then satisfies $A \subseteq A_R$ and $|A_R \setminus A| \le \varepsilon(N - n)$, so

$$|A_R| \;\le\; n + \varepsilon(N - n) \;\approx\; n + \varepsilon N \quad (N \gg n). \tag{8.7}$$

The verdict on x depends on R only through which side of A_R the query lands, so A_R is the representation-independent object that does the work.

The theorem

Theorem 8.3.1 (Entropy floor for FNR-zero approximate sets)**.** *Let (R, query) be an FNR-zero representation scheme of bit length L that, for every n-element $A \subseteq U$, produces $A_R \supseteq A$ with $|A_R \setminus A| \le \varepsilon(N - n)$. Then in the regime $N \gg n$ with $\varepsilon \in (0, 1)$ bounded away from 1,*

$$L \;\ge\; n \log_2(1/\varepsilon) - o(n), \qquad \textit{equivalently} \qquad L/n \;\ge\; -\log_2 \varepsilon + o(1). \tag{8.8}$$

Proof. The argument is a pigeonhole count. Fix the accepted set A_R of a representation R. By FNR-zero $A \subseteq A_R$, and by (8.7) $|A_R| \le n + \varepsilon N$, so the true set is some n-subset of A_R: any of $\binom{|A_R|}{n}$ choices is consistent with what R reports. The algorithm cannot distinguish among them, so R describes an equivalence class of that size, not a single A. Every one of the $\binom{N}{n}$ possible target sets needs some $R \in \{0,1\}^L$ whose accepted set contains it, and there are at most 2^L strings, each covering at most $\max_{A_R} \binom{|A_R|}{n}$ sets, so

$$2^L \cdot \max_{A_R} \binom{|A_R|}{n} \;\ge\; \binom{N}{n}. \tag{a}$$

Take logarithms, bound $|A_R| \le n + \varepsilon N$, and apply Stirling: $\log_2 \binom{N}{n} = n \log_2(N/n) + O(n)$ and, in the regime $\varepsilon N \gg n$, $\log_2 \binom{n+\varepsilon N}{n} = n \log_2(\varepsilon N/n) + O(n)$. Subtracting,

$$L \;\ge\; n \log_2(N/n) - n \log_2(\varepsilon N/n) - O(n) \;=\; n \log_2(1/\varepsilon) - O(n),$$

and dividing by n gives $L/n \ge -\log_2 \varepsilon + o(1)$. The full asymptotic control of the $o(n)$ term is in Towell [Tow26h]. □

The simplification $\varepsilon N \gg n$ is the deployment regime: ε fixed, N large relative to n, the target embedded in a much larger universe. There the floor is sharp; when U is barely larger than A the bound holds only in the rougher form (a) and the per-element reading degrades. (At the other extreme $\varepsilon = 1$ the floor is 0, met by the trivial always-accept structure that stores nothing.)

Form of the bound

The bound has no parameters to tune: the per-element cost is a logarithm of the inverse error rate, independent of how the representation is built.

ε	$-\log_2 \varepsilon$ (bits/element)	bits for $n = 10^6$
10^{-1}	≈ 3.32	$\approx 0.42\,\text{MB}$
10^{-2}	≈ 6.64	$\approx 0.83\,\text{MB}$
10^{-3}	≈ 9.97	$\approx 1.25\,\text{MB}$
10^{-6}	≈ 19.93	$\approx 2.49\,\text{MB}$
10^{-9}	≈ 29.90	$\approx 3.74\,\text{MB}$

A million-element FNR-zero set at one-in-a-billion false-positive rate cannot fit in less than about 3.74 megabytes, whatever the encoding. The cost grows *additively*, by $\log_2 10 \approx 3.32$ bits per element for each order of magnitude in $1/\varepsilon$, not geometrically. The universe size N does not enter the leading term: one might have expected the $\log_2 \binom{N}{n}$ cost of identifying A exactly, but the FPR budget buys back precisely that universe-dependent excess, leaving the residual $n \log_2(1/\varepsilon)$.

Why the floor is one-sided

Theorem 8.3.1 is asymmetric because its constraint is. The pigeonhole step (a) uses FNR-zero to force $A \subseteq A_R$, and that containment is what makes $\binom{|A_R|}{n}$ the right equivalence-class count. If the false-negative rate were also positive, a true element could be misclassified, A would no longer sit inside A_R, and the containment the proof rests on would break.

The two-sided lower bound is, at present, an open problem. A natural heuristic keeps a $\tau = 1 - \omega$ fraction of the n elements on average, each still costing $-\log_2 \varepsilon$ bits, which suggests a cost near $-(1-\omega)\, n \log_2 \varepsilon$; but this is *not* a valid information-theoretic lower bound. Randomly dropping elements changes the encoding problem qualitatively rather than merely scaling it, and the pigeonhole count no longer applies. What can be said is only qualitative: a two-sided set may in principle be represented below the FNR-zero floor $n \log_2(1/\varepsilon)$, spending part of its error budget on false

negatives to blur the identity of A, but no proven bound quantifies how far below. The FNR-zero floor is the rigorous result, and it is the one the rest of the book builds on.

The constructive question

Theorem 8.3.1 forbids: no FNR-zero scheme beats $-\log_2 \varepsilon$ bits per element. Whether the floor is achievable is the dual question. The standard Bloom filter pays a factor of about 1.44, derived in Section 8.4; chapter 14 (Chapter 14) constructs the Bernoulli Hash Function, which meets the floor asymptotically. The two together characterize the FNR-zero family tightly: the floor sits at $-\log_2 \varepsilon$, and a known scheme reaches it. Until then the floor is a target, and Section 8.4 measures how far the standard Bloom filter sits above it and why.

8.4 The Bloom Filter's Gap

The standard Bloom filter is FNR-zero (Definition 5.2.1), so Theorem 8.3.1 governs it. How far above the floor does it sit? The answer is a single constant, $\log_2 e \approx 1.443$, the same at every ε.

Bits per element

By Theorem 5.2.1 the optimal Bloom filter is half-occupied with

$$\varepsilon^* \;=\; 2^{-(m/n)\ln 2}. \tag{8.9}$$

Inverting at $\varepsilon^* = \varepsilon$, $\log_2 \varepsilon = -(m/n)\ln 2$, so

$$\frac{m}{n} \;=\; \frac{\log_2(1/\varepsilon)}{\ln 2} \;=\; \log_2(1/\varepsilon)\cdot \log_2 e, \tag{8.10}$$

using $1/\ln 2 = \log_2 e$. The Bloom budget is thus a constant multiple $\log_2 e$ of the floor cost $\log_2(1/\varepsilon)$ of Theorem 8.3.1, so the ratio is $\log_2 e = 1/\ln 2 \approx 1.4427$, independent of ε.

Proposition 8.4.1 (Bloom filter overhead)**.** *At every $\varepsilon \in (0,1)$ the optimal standard Bloom filter uses $m/n = \log_2(1/\varepsilon)\cdot \log_2 e$ bits per element, which is $\log_2 e \approx 1.4427$ times the floor of Theorem 8.3.1. The relative overhead is constant.*

The constant is the chapter-5 half-occupancy result read dually: $\ln 2 = 1/\log_2 e$ is the optimal load that maximizes per-bit entropy, and the same $\log_2 e$ is the gap to the floor. Half-occupancy is information-theoretically the wrong target, but it is the best a representation built from k independent hashes and a flat bit array can do.

The gap in absolute terms

A constant factor on a function that grows like $\log_2(1/\varepsilon)$ produces an absolute gap that grows with it:

ε	floor (bits/el)	Bloom (bits/el)	gap (bits/el)
10^{-2}	6.64	9.59	2.95
10^{-3}	9.97	14.38	4.41
10^{-6}	19.93	28.76	8.83
10^{-9}	29.90	43.14	13.25

At one-percent FPR the filter pays about three extra bits per element; at one-in-a-million, about nine. For a million-element set at $\varepsilon = 10^{-6}$ the gap is 8.83 megabits, about 1.10 megabytes, that the standard construction cannot recover. A constant factor on an already-small quantity is acceptable in many settings, and Bloom filters dominate the ecosystem for good reason; the point of Proposition 8.4.1 is to quantify the slack, not to condemn the construction, and to ask which design choice forces it.

Why the gap is structural

The Bloom filter pays its overhead because of how it is built: k independent hashes write into a single flat bit array, and a query accepts iff all k read bits are 1. The optimal-k analysis sets the per-bit fill to $p = 1/2$, where the array carries one full bit of Shannon entropy per array bit and the encoding is maximally dense at the bit level. But the floor is set by the membership problem, not by the encoding's internal density. The Bloom filter maximizes per-bit entropy and takes the kth power of it, getting a clean optimum at half-occupancy while spending bits on the array's internal balance rather than on the discrimination it was meant to perform.

Remark (Compressed Bloom filters)**.** *Half-occupancy is exactly what makes the optimal array* incompressible*: at $p = 1/2$ each bit carries a full bit of entropy, so arithmetic coding of the optimal-k array recovers nothing. Mitzenmacher [Mit02] therefore goes the other way, using a* sparser *array (fewer hashes, fill $p \neq 1/2$) whose low per-bit entropy compresses well, trading a*

larger in-memory array and the loss of in-place random-access queries for a smaller transmitted size. The compressed cost per element is $\log_2(1/\varepsilon) \cdot g(p)$ *with* $g(p) = \frac{-p \ln p - (1-p)\ln(1-p)}{\ln p \, \ln(1-p)}$*: at* $p = 1/2$*,* $g = 1/\ln 2 \approx 1.44$ *(the uncompressed overhead, no gain), while* $g(p) \to 1$ *as* $p \to 0$*, so the sparse limit* approaches *the floor as* $\varepsilon \to 0$ *without reaching it at finite* ε*. Compression recovers the per-bit redundancy of a non-half-full array; the residual gap is structural, and closing it requires abandoning the flat-array-with-k-hashes representation entirely.* △

That is what chapter 14 does. The Bernoulli Hash Function of Chapter 14 replaces the flat array with a perfect-hash-based encoding: each element gets a deterministic location from a perfect hash, and a fingerprint of $\log_2(1/\varepsilon)$ bits is stored there, so collisions in the fingerprint table are the only source of false positives and the $\log_2 e$ overhead vanishes asymptotically. The gap is not a deficiency of analysis but a consequence of a specific structural choice, and a different choice closes it. Section 8.5 sketches the route there.

8.5 What Comes Next

Part II is complete. From the approximate set as a noisy bit per universe element (Chapter 5) it developed the algebra of unions, intersections, and complements (Chapter 6), the estimation and diagnostic discipline for the two rates (Chapter 7), and the information-theoretic floor that pins the per-element space cost of any FNR-zero representation (this chapter). The atom is now fully described: its rates are estimable, its compositions predictable, its space cost bounded below by $-\log_2 \varepsilon$ (Theorem 8.3.1), and the standard Bloom filter sits a known $\log_2 e$ above that floor (Section 8.4).

Part III generalizes from sets to maps: Chapter 9 lifts Bernoulli[bool] to Bernoulli[T] for arbitrary algebraic types, where an approximate map into a k-element output is a $k \times k$ channel rather than a single (ε, ω) pair, and chapters 10 through 13 develop sketches, randomized algorithms, approximate relations, and the formal type framework. Part IV builds the constructive payoff: Chapter 14 constructs the Bernoulli Hash Function that meets the entropy floor asymptotically by abandoning the flat bit array, and Chapter 15 builds the FPR-zero counterpart on the other one-sided axis. Part V draws the boundary: Chapter 16 catalogues the systems that are *not* Bernoulli, where errors correlate across inputs, rates drift, or adversarial queries defeat uniform sampling, and the closing chapters turn the model's controlled error into a cryptographic resource. The thread is one

throughout: an atom of controlled error, its algebra, its floor, its optimal constructions, and the boundary that fences off what the atom cannot describe.

Bibliographic Notes

The entropy manuscript. The floor of Theorem 8.3.1 and the Bloom overhead of Proposition 8.4.1 follow Towell [Tow26h], the primary source for Section 8.3 and Section 8.4: the counting argument, the asymptotic $-n \log_2 \varepsilon$ form, and the $\log_2 e$ comparison are all from it, as is the honest statement that the two-sided case remains open. This chapter adds the BSC bridge of Section 8.2 that ties the floor to Part I's running channel, the entropy refresher for readers new to $H(X)$, and the explicit forward connection to the chapter-14 construction that closes the gap.

The counting argument. The proof of Theorem 8.3.1 is the classical lower bound of Carter, Floyd, Gill, Markowsky, and Wegman, in the combinatorial style of the universal hashing program of Carter and Wegman [CW79]: reason about a hash-based data structure by counting collisions over an explicit family rather than appealing to a random oracle. For every representation R, count the accepting sets it can describe; for every target set, count the representations that work; the ratio is the bound.

Entropy and capacity. The background of Section 8.1 and Section 8.2 is standard; Cover and Thomas [CT06] covers entropy, mutual information, the BSC, capacity, and the source-coding theorem in its first three chapters, with the BSC capacity $C = 1 - H(\varepsilon)$ and the source-coding bound that underwrites Theorem 8.3.1 treated rigorously. Shannon [Sha48] remains the cleanest short presentation of the source-coding idea.

Compressed Bloom filters. Mitzenmacher [Mit02] introduced the compressed Bloom filter, whose mechanism Section 8.4 works out: the half-full optimal array is incompressible, so compression instead exploits a deliberately sparse array, recovering per-bit redundancy and approaching the floor for the *transmission* setting it targets, while the in-place query structure the rest of the chapter analyzes keeps the $\log_2 e$ gap. It is the right entry point for how much of that gap compression alone can close, and sets up the contrast with the structural change of Chapter 14. Mitzenmacher and Upfal [MU17] is the companion for the probabilistic background.

Part III

From Sets to Maps

Chapter 9

Approximate Functions and n-ary Channels

9.1 From Approximate Sets to Approximate Functions

Part II read a classical set as its membership predicate $\mathbb{1}_A : U \to \{\top, \bot\}$ and made the predicate noisy: each evaluation $\mathsf{A}^{\pm}(x)$ is a Bernoulli[bool] instance, and the chapter-4 axioms collapse the per-element rates to a single pair (ε, ω). The framing pinned the codomain to $\{\top, \bot\}$. Nothing in it requires two values, only that each evaluation draws from a distribution on the codomain fixed by the input's block. Replacing $\{\top, \bot\}$ with any type T gives the *approximate function*

$$\tilde{f} : U \longrightarrow T,$$

whose evaluation $\tilde{f}(x)$ is an instance of the lifted type Bernoulli[T], the T-valued analogue of the noisy bit. For $T = \text{bool}$ this is exactly the membership predicate of an approximate set, so Part II is the special case; for $T = \{1, \ldots, K\}$ it is a K-way classifier; for $T = \mathbb{N}$ it is a counter or frequency estimator. The two-rate description gives way to a full per-block output distribution, and the 2×2 channel matrix of Part II becomes a $K \times L$ matrix with K input blocks, $L = |T|$ output values, rows summing to one, and (i, j)-entry the probability that block i produces output j.

Two examples. A 3-class classifier on $T = \{\text{cat}, \text{dog}, \text{bird}\}$ has its input domain split into three blocks by true class, with per-block behavior a 3×3

confusion matrix: diagonal entries the per-class accuracies, off-diagonals the confusion rates, six free parameters once rows are normalized. A counter on $T = \mathbb{N}$ has input blocks B_n indexed by true count n, and row n is a distribution on $\mathbb{N}$ concentrated near n with a tail width that records the counter's precision; the matrix is countably infinite but each row is still a probability distribution. In both cases the structure is the same: a function from inputs to a typed codomain, with the per-query distribution fixed by the input's block.

What transfers. The two chapter-4 axioms carry over unchanged, since both are independence claims indifferent to alphabet size: Axiom 1 keeps error events at distinct inputs independent, Axiom 2 keeps each per-block distribution stable. The parameter count grows from 2 to $K(L-1)$ (each of K rows is a distribution on L outputs). Two structural results scale accordingly. The Kronecker factorization of Theorem 4.4.1 (Section 4.4), which factored the joint of m Boolean queries as a tensor product of 2×2 matrices, holds with $K \times L$ matrices in their place (Section 9.3). The BSC capacity $C = 1 - H(\varepsilon)$ of Proposition 8.2.1 (Section 8.2) is specific to the binary case, but capacity as the maximum mutual information over input distributions generalizes, with closed forms for the symmetric sub-family and numerical recipes otherwise (Section 9.4). Section 9.2 gives the formal definition first.

9.2 N-ary Channels

Section 9.1 described an approximate function $\tilde{f} : U \to T$ whose per-block behavior is a distribution on T. This section gives the formal object: a row-stochastic matrix, with the binary 2 of Part II replaced by the number of input blocks K and the output cardinality $L = |T|$.

Definition 9.2.1 (n-ary channel)**.** *An* n-ary channel *is a triple* (X, Y, Q) *where* X *is an input domain partitioned into* K *blocks* $B_1, \ldots, B_K$ *(inputs with the same output distribution share a block, one per row of* Q*),* Y *is an output domain with* $|Y| = L$*, and* Q *is a* $K \times L$ *row-stochastic matrix with* $Q_{ij} \geq 0$*,* $\sum_j Q_{ij} = 1$*, and*

$$Q_{ij} \;=\; \mathrm{P}\left(\mathit{output} = j \middle| \mathit{input} \in B_i\right).$$

The triple satisfies Axiom 1 (Axiom 1: error events at distinct inputs are independent) and Axiom 2 (Axiom 2: the per-block output distributions are

conditionally independent given the objective function). Setting $K = L = 2$ recovers Definition 4.5.1.

Model order versus row count. The integer K counts the rows of Q, one per input block. The *model order* of Definition 2.4.1 is a different and generally smaller count: the number of blocks under the coarsest partition that merges inputs sharing a common *equality-error rate* $\varepsilon(x) = \mathrm{P}\left(\tilde{f}(x) \neq x\right)$. The two coincide when every block has a distinct error rate, and the model order is strictly smaller when blocks share one. The distinction matters at the very first example. The binary *symmetric* channel writes as a 2×2 matrix ($K = 2$ rows) but is model order *one*: both inputs flip at the common rate ε, so they form a single error-rate block (the first-order Bernoulli model of Section 2.4). The binary *asymmetric* channel is also 2×2 but model order two, since its inputs carry different error rates $\omega \neq \varepsilon$. Throughout this chapter K is the matrix row count; we name the model order only where the distinction is in play.

Notation: K uppercase, k lowercase. The uppercase K (input-block count) and L (output cardinality) are structural parameters of the channel. The lowercase k is reserved for the tuning knobs of Parts I and II, the Bloom-filter hash count (Section 5.2) and the Miller-Rabin round count (Section 3.4); the case distinction keeps the structural from the tunable. A Bloom filter has $K = 2$ input blocks (members and non-members) and hash count k; both blocks of a Bloom filter or a Miller-Rabin test carry different error rates (members never miss, non-members false-positive at ε), so these are genuinely model order two, while their lowercase k is a design choice.

Examples

Binary, symmetric and asymmetric. The BSC of Definition 2.1.1 is $Q_{\mathrm{BSC}} = \left(\begin{smallmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{smallmatrix}\right)$: $K = 2$ rows, $L = 2$, model order one. Dropping the symmetry recovers the asymmetric channel of Definition 2.2.1, $Q_{\mathrm{asym}} = \left(\begin{smallmatrix} 1-\omega & \omega \\ \varepsilon & 1-\varepsilon \end{smallmatrix}\right)$, with two independent rates and model order two; the Z-channel is the one-sided specialization $\omega = 0$.

A 3-class classifier ($K = L = 3$). A confusion matrix is an n-ary channel. A classifier over $\{\text{cat}, \text{dog}, \text{bird}\}$ has input blocks by true class and

$$Q = \begin{pmatrix} 0.90 & 0.05 & 0.05 \\ 0.05 & 0.90 & 0.05 \\ 0.05 & 0.05 & 0.90 \end{pmatrix},$$

rows summing to one, diagonal the per-class accuracies, off-diagonals the confusion rates. A generic 3-class classifier has three distinct per-class error profiles, model order three, and $K(L-1) = 3 \cdot 2 = 6$ free parameters. The displayed matrix is more constrained: all three classes share the equality-error rate 0.10 with an even off-diagonal split, so it is model order *one*, specified by the single number 0.90. The matrix dimension (3×3) and the model order (1) are again different counts.

A counter ($K = L = \infty$). A frequency estimator on $T = \mathbb{N}$ has input blocks B_n indexed by true count n and an infinite channel matrix whose row n is a distribution on $\mathbb{N}$ concentrated near n. CountMin gives a one-sided tail (it over-counts but never under-counts), HyperLogLog a relative-error tail; chapter 10 builds the constructions.

A physical instance: quantum readout. The n-ary channel has a direct physical realization in quantum computing. Measuring n qubits in the computational basis is a channel whose $2^n \times 2^n$ row-stochastic matrix $Q_{ij} = \mathrm{P}(\text{observed bitstring } j | \text{prepared classical state } i)$ is the *assignment matrix* that readout-error mitigation calibrates and inverts [NC10]. When per-qubit readout errors are independent, Q factors as a Kronecker product of 2×2 single-qubit matrices, the n-ary Kronecker factorization of Section 9.3, and each single-qubit matrix is asymmetric in practice (energy relaxation makes $|1\rangle \to |0\rangle$ misreads likelier than the reverse), the Z-channel-skewed asymmetric channel above. The correspondence is exact only on the diagonal: Q presupposes a definite prepared state, which a pre-measurement superposition lacks, and the channel omits the interference such a state carries. The Bernoulli channel is in this sense the classical, decohered shadow of a quantum measurement; Section 16.5 marks where the shadow stops being faithful.

Axiom transfer and parameter count

Both chapter-4 axioms lift unchanged, being independence claims indifferent to alphabet size: Axiom 1 keeps the events $\{\text{output on } x_1 = j_1\}$ and

{output on $x_2 = j_2$} independent for $x_1 \neq x_2$, and Axiom 2 keeps the K row distributions of Q conditionally independent given the objective function. Together they make Q a sufficient summary, as (ε, ω) was in the Boolean case.

A $K \times L$ row-stochastic matrix has KL entries and K row-sum constraints, leaving

$$K(L-1)$$

free parameters, each row a distribution on L outputs with $L-1$ degrees of freedom. This is the n-ary generalization of Corollary 4.4.1.2, recovering the (ε, ω) pair at $K = L = 2$. The count is an upper bound: symmetric structure cuts it further (the BSC's shared rate drops it from 2 to 1, the symmetric 3-class matrix above from 6 to 1). The contrast with an unconstrained joint over $K \times L$ outcomes, which carries $KL - 1$ parameters, is the same parsimony Axioms 1 and 2 bought in chapter 4. Section 9.3 lifts the Kronecker theorem and Section 9.4 the capacity formula, both with the binary 2 replaced by L.

9.3 The n-ary Kronecker Theorem

At model order two, chapter 4 proved that the joint confusion matrix of m queries on a Bernoulli predicate factors as the Kronecker product of the m per-query matrices (Theorem 4.4.1), with the unconditional form holding in the deterministic-rates regime (Corollary 4.4.1.1) that every worked example in the book inhabits. The same factorization holds for n-ary channels, with the binary output alphabet replaced by one of size L. We drop the conditional star and write Q for the deterministic per-query matrix.

Theorem 9.3.1 (n-ary Kronecker factorization)**.** *Let (X, Y, Q) be an n-ary channel with K input blocks and output cardinality L, satisfying Axiom 1 and Axiom 2 in the deterministic-rates regime. For m queries on distinct inputs $X_1, \ldots, X_m$ with $X_i \in B_{j_i}$ and outputs $Y_1, \ldots, Y_m$,*

$$\mathrm{P}\left(Y_1, \ldots, Y_m | X_1 \in B_{j_1}, \ldots, X_m \in B_{j_m}\right) \;=\; \prod_{i=1}^{m} Q_{j_i, Y_i},$$

so the $K^m \times L^m$ joint channel matrix factors as the m-fold Kronecker power $Q^{(m)} = Q^{\otimes m}$.

Proof. The deterministic-rates regime fixes each per-block output distribution as a row of Q, and Axiom 2 makes the distribution of Y_i given $X_i \in B_{j_i}$

equal to row j_i regardless of the other queries. The outputs are deterministic functions of the per-input error events, which Axiom 1 makes mutually independent across distinct inputs, so the joint factors as

$$\mathrm{P}\left(Y_1 = y_1, \ldots, Y_m = y_m | X_i \in B_{j_i} \text{ for each } i\right) = \prod_{i=1}^{m} \mathrm{P}\left(Y_i = y_i | X_i \in B_{j_i}\right) = \prod_{i=1}^{m} Q_{j_i, y_i},$$

which is by definition the $(j_1 \cdots j_m,\, y_1 \cdots y_m)$ entry of $Q^{\otimes m}$. The only binary-specific step in chapter 4 was reading each row as a (ε, ω) pair; here each row is a length-L probability vector and the rest is identical. □

Queries on *different* elements that fall in the same block stay independent under Axiom 1; only the row of Q they consult is shared.

Corollary 9.3.1.1 (Parameter count for joint n-ary queries)**.** *The joint matrix $Q^{(m)}$ is fixed by the single $K \times L$ matrix Q, which carries $K(L-1)$ free parameters; adding queries adds no parameters. An unconstrained m-query joint channel carries $K(L^m - 1)$ parameters, growing exponentially in m.*

Proof. $Q^{\otimes m}$ is determined entirely by Q, whose rows are K distributions on L outputs ($K(L-1)$ free parameters). A new query position appends another factor of the same Q to the product. The unconstrained channel assigns each of the K source blocks a free distribution on the L^m joint outputs, $L^m - 1$ parameters each; the ratio $\frac{K(L^m-1)}{K(L-1)} = \frac{L^m-1}{L-1}$ grows exponentially in m. □

At $K = L = 2$ this recovers Corollary 4.4.1.2: $K(L-1) = 2$ and $K(L^m - 1) = 2(2^m - 1)$. The reduction sharpens at larger L: a 10-output channel queried 50 times is fixed by $K \cdot 9$ numbers rather than $K(10^{50} - 1)$.

Remark (Serial composition is matrix multiplication)**.** *The Kronecker power composes m* parallel *queries of one channel. Two channels in* series *compose differently. If Q_1 is $K_1 \times L_1$ (input blocks to an intermediate alphabet) and Q_2 is $L_1 \times L_2$ (intermediate to output), the composite is the matrix product*

$$Q \;=\; Q_1 Q_2, \qquad Q_{ik} \;=\; \sum_j (Q_1)_{ij} (Q_2)_{jk},$$

the probability that input block i produces output k after both stages. Each row of Q sums to one ($\sum_k \sum_j (Q_1)_{ij}(Q_2)_{jk} = \sum_j (Q_1)_{ij} \cdot 1 = 1$), so Q is again an n-ary channel: a multi-stage approximate pipeline, a coarse classifier feeding a fine one, is itself one channel in the sense of Definition 9.2.1.

Serial composition is matrix product, parallel querying is Kronecker product; the two are the n-ary face of the function-composition and product-type structure Part III develops. △

Worked example

Take the 3-class matrix Q above and three queries on blocks (B_1, B_2, B_1). The all-correct output $(1, 2, 1)$ has joint probability

$$Q_{1,1}Q_{2,2}Q_{1,1} = 0.9 \cdot 0.9 \cdot 0.9 = 0.729,$$

the n-ary analogue of the BSC's $(1-\varepsilon)^m$ joint success. Misclassifying the first query, output $(2, 2, 1)$, gives

$$Q_{1,2}Q_{2,2}Q_{1,1} = 0.05 \cdot 0.9 \cdot 0.9 = 0.0405.$$

The joint matrix $Q^{(3)}$ has shape 27×27; the unconstrained joint parameter count $K(L^m - 1) = 3(27 - 1) = 78$ collapses to the $K(L-1) = 6$ that specifies Q, and at $m = 10$ the gap is $3(3^{10} - 1) \approx 1.77 \times 10^5$ against the same 6. Section 9.4 turns from structure to the per-use information bound.

9.4 Capacity for n-ary Channels

The BSC capacity $C = 1 - H(\varepsilon)$ of Proposition 8.2.1 is the $K = L = 2$ case of a definition that lifts to any $K \times L$ channel. The lift is mostly definitional; what changes is that the optimization no longer has a single closed form except for a symmetric sub-family.

The general case

For an n-ary channel (Definition 9.2.1), an input distribution P_X on the K blocks induces $P_Y(j) = \sum_i P_X(B_i)Q_{ij}$, and the capacity (8.5) is

$$C \;=\; \max_{P_X} \big[H(Y) - H(Y \mid X)\big], \qquad H(Y \mid X) = \sum_{i=1}^{K} P_X(B_i)\, H(Q_{i\cdot}), \tag{9.1}$$

the same object as the BSC's, now over the $(K-1)$-simplex rather than the interval $[0, 1]$. The mutual information is concave in P_X, so the maximum exists, but the maximizer satisfies implicit stationarity conditions with no closed form in general. The *Blahut–Arimoto* algorithm, an alternating fixed-point iteration over P_X and the posterior $\mathrm{P}(X \mid Y)$, converges geometrically to the capacity-achieving distribution (Cover and Thomas [CT06, Ch. 10]), making the general n-ary capacity computable to any precision.

Remark (The output-entropy ceiling)**.** *Every n-ary channel obeys* $C \leq \log_2 L$*: since* $H(Y \mid X) \geq 0$*,* $C \leq \max_{P_X} H(Y) \leq \log_2 L$*, the maximum entropy of an* L*-symbol alphabet. Equality requires both* $H(Y \mid X) = 0$ *(the channel is a noise-free function of the input) and an input distribution making* P_Y *uniform, that is, a deterministic surjection from input blocks onto the* L *outputs. This is the n-ary ceiling that* $\log_2 2 = 1$ *bit was for the BSC.* △

Symmetric channels

The BSC's closed form rested on two facts: its rows are permutations of one another (so $H(Y \mid X)$ is input-independent) and its columns are too (so uniform input gives uniform output). Lifting both gives a sub-family with a closed-form capacity.

Call a channel *weakly symmetric* if every row of Q is a permutation of a common multiset $\mathbf{r} = (r_1, \ldots, r_L)$ *and* every column has the same sum. The first condition makes $H(Y \mid X) = H(\mathbf{r})$ input-independent; the second makes the uniform input $P_X(B_i) = 1/K$ produce the uniform output $P_Y(j) = 1/L$, attaining $H(Y) = \log_2 L$. (Rows being permutations of one another does *not* alone uniformize the output: the column sums must match.) Hence

$$C \;=\; \log_2 L \;-\; H(\mathbf{r}), \tag{9.2}$$

the n-ary analogue of $C = 1 - H(\varepsilon)$: the ceiling $\log_2 L$ less the irreducible per-use row uncertainty $H(\mathbf{r})$.

A *strongly symmetric* channel has, in addition, columns that are permutations of one another. The clean parametrization spreads a single error probability p_e uniformly over the $L - 1$ wrong outputs,

$$Q_{ij} = \begin{cases} 1 - p_e & j = i, \\ p_e/(L-1) & j \neq i, \end{cases}$$

a square $K = L$ matrix with the BSC as its $L = 2$, $p_e = \varepsilon$ instance.

Proposition 9.4.1 (Strongly symmetric n-ary capacity)**.** *A strongly symmetric channel with output size* L *and per-input error* $p_e \in [0, 1)$ *has capacity*

$$C \;=\; \log_2 L \;-\; H\Big(1 - p_e,\ \tfrac{p_e}{L-1},\ \ldots,\ \tfrac{p_e}{L-1}\Big) \quad \textit{bits per use,} \tag{9.3}$$

attained at the uniform input.

At $L = 2$, $p_e = \varepsilon$, the row $(1-\varepsilon, \varepsilon)$ has entropy $H(\varepsilon)$ and $C = 1 - H(\varepsilon)$, recovering (8.6). The endpoints carry over: $p_e = 0$ gives $C = \log_2 L$ (noiseless L-ary channel), and $p_e = (L-1)/L$ gives the uniform row and $C = 0$.

Worked example

Take $L = 4$, $p_e = 0.1$, so each off-diagonal entry is $0.1/3 \approx 0.0333$ and the channel is strongly symmetric. The ceiling is $\log_2 4 = 2$ bits. The row entropy is

$$H(\mathbf{r}) = -0.9 \log_2 0.9 - 3 \cdot \tfrac{0.1}{3} \log_2 \tfrac{0.1}{3} \approx 0.137 + 0.490 = 0.627 \text{ bits},$$

so $C = 2 - 0.627 \approx 1.373$ bits per use, well below the noiseless ceiling of 2 and well above the BSC's 0.531 at the same per-symbol error 0.1. The gap to the ceiling shrinks as L grows at fixed p_e: the error mass spreads more thinly and the row entropy grows sublinearly. Section 9.5 previews how concrete sketches, each an n-ary channel with its own Q and its own symmetry, instantiate the framework.

9.5 Bridge to Sketches

A *sketch* maps a stream or multiset to a compact summary, often sublinear in the input size, from which a query routine reconstructs an answer with error bounded by an analytic expression in the summary size: the user trades a known accuracy for a known space budget. What distinguishes a sketch from lossy compression is that bounded, calculable error. The three canonical sketches, CountMin for frequency, HyperLogLog for cardinality, and MinHash for similarity, are each a Bernoulli[T] instance of this chapter.

Each is a channel in the sense of Definition 9.2.1, differing only in the structure of Q. CountMin is a counter ($T = \mathbb{N}$) whose one-sided overestimation puts all of Q's mass at or above the diagonal; HyperLogLog is a counter whose mass concentrates in a multiplicative band around the diagonal (relative rather than one-sided error); MinHash is a similarity estimator ($T = [0, 1]$) whose Q is a continuous kernel with output concentrated in a band of width $O(1/\sqrt{r})$ at sketch size r. All three share the same skeleton, an input type, an output type, a row-stochastic kernel, and the per-query independence of Axioms 1 and 2 (Axiom 1, Axiom 2); each one's error bound is a statement about the entries of Q, its composition rule a statement

about how Q-matrices combine, and its capacity a specialization of Proposition 9.4.1. Chapter 10 builds the three from scratch, the first non-Boolean instances of Bernoulli[T] the book treats in detail.

Bibliographic Notes

The approximate-maps manuscript. The lift of the noisy bit Bernoulli[bool] to the type-general Bernoulli[T], with the n-ary channel matrix Q organizing its per-block output distributions, follows Towell [Tow26l]: the T-valued framing of Section 9.1, the $K \times L$ channel of Definition 9.2.1, the Kronecker factorization Theorem 9.3.1 with its parameter count Corollary 9.3.1.1, and the capacity formulation of Section 9.4 are from it. This chapter recalls Section 4.4 and Section 8.2 as the model-order-two special cases the n-ary results generalize, and exhibits the 3-class classifier and the counter as concrete non-Boolean instances.

N-ary channels and Blahut–Arimoto. The discrete memoryless channel is standard, with Cover and Thomas [CT06] the depth reference: their chapter 7 defines capacity as the supremum of mutual information over input distributions and treats the symmetric and weakly symmetric special cases whose closed forms Section 9.4 uses, and their chapter 10 develops the Blahut–Arimoto algorithm, the coordinate ascent on the mutual information that makes the general n-ary capacity of Proposition 9.4.1 computable rather than merely defined.

Shannon's framework. The channel-capacity vocabulary originates in Shannon [Sha48]: the entropy of Section 8.1, the capacity $C = \max_{P_X} I(X;Y)$ of Section 9.4, and the source-coding theorem connecting them. Shannon's channel as a probabilistic map from input to output alphabet under a fixed transition kernel is exactly the framing of Definition 9.2.1: Q is his transition matrix, K and L his alphabet sizes, the chapter's capacity his capacity. The sketch literature previewed in Section 9.5 is taken up with its citations in Chapter 10, where the named constructions belong, rather than with the abstract framework that subsumes them.

Chapter 10

Sketches and Estimators

10.1 Sketches as Bernoulli of T

A *sketch* maps a stream or multiset over a universe U to a compact summary, with a query routine that reconstructs an estimate $\hat{\theta}$ of some quantity θ of the stream from the summary alone. What makes it a sketch, rather than a compressed representation, is that the summary grows sublinearly (often poly-logarithmically) in the stream length, the per-item update is near-constant, and the error $\hat{\theta} - \theta$ is bounded by a clean function of the design parameters (Section 9.5). This chapter builds three, distinguished by what they estimate, which is to say by their output type.

CountMin (Section 10.2) estimates per-item frequency f_a, with $T = \mathbb{N}$ and *one-sided* error: the construction guarantees $\hat{f}_a \geq f_a$ deterministically, the multiset analogue of the Bloom filter's one-sided membership report (Chapter 5). *HyperLogLog* (Section 10.3) estimates distinct-item cardinality n, again with $T = \mathbb{N}$ but with two-sided relative error of order $1/\sqrt{m}$ in the register count m. *MinHash* (Section 10.4) estimates Jaccard similarity $J(A, B)$, with $T = [0, 1]$ and symmetric error of variance $J(1 - J)/k$ in the signature length k.

The chapter-9 framework gives all three the same shape. Each is a function $s : U^* \to T$ whose error is summarized by a row-stochastic channel Q (Definition 9.2.1) with rows indexed by the true value θ, columns by the report $\hat{\theta}$: CountMin's Q is upper triangular (one-sided), HyperLogLog's concentrates in a multiplicative band (relative error), MinHash's is binomial along each row. The structural payoff is composition: the per-sketch independence assumptions all reduce to the same channel statement, and merging two sketches or querying one at k positions is the Kronecker factor-

ization of Theorem 9.3.1 rather than a per-sketch lemma. Each of the next three sections follows one template, construction, error bound, Bernoulli[T] type, merge; Section 10.5 hands the pattern to the type algebra of Part III's close.

10.2 CountMin

A data stream presents items $x_1, \ldots, x_n$ from a universe U with repetition. For $a \in U$, write $f_a = |\{i : x_i = a\}|$ for its true frequency and $F = \sum_b f_b = n$ for the stream length. The frequency-estimation query reports an estimate $\hat{f}_a$ of f_a in a small structure of size polylogarithmic in F, with a tunable error bound.

The construction

Fix a width w and depth d and pick d hash functions $h_1, \ldots, h_d : U \to \{1, \ldots, w\}$ from a pairwise-independent family ($\mathrm{P}\,(h_i(a) = u,\, h_i(b) = v) = 1/w^2$ for distinct a, b), which is cheaper than the full independence the Bloom filter assumed (Definition 5.2.1).

Definition 10.2.1 (CountMin sketch)**.** *The* CountMin sketch *of parameters (w, d) is a $d \times w$ integer table C, initialized to zero, with d hashes, and two operations:* Update(a) *increments $C[i][h_i(a)]$ for each i, and* Query(a) *returns $\hat{f}_a = \min_i C[i][h_i(a)]$.*

Algorithm 3: CountMin update and query

```
1  procedure Update(a)
2  |   for i ← 1 to d do
3  |   |   C[i][h_i(a)] ← C[i][h_i(a)] + 1;
4  |   end
5  function Query(a)
6  |   m ← ∞;
7  |   for i ← 1 to d do
8  |   |   if C[i][h_i(a)] < m then m ← C[i][h_i(a)];
9  |   end
10 |   return m;
```

Both operations run in $O(d)$ time, independent of F and $|U|$, in $d \cdot w$ counters. Each cell $C[i][h_i(a)]$ holds f_a plus a non-negative collision mass

from other items hashing to the same cell, so $C[i][h_i(a)] \geq f_a$ for every i and $\hat{f}_a = \min_i C[i][h_i(a)] \geq f_a$. The estimate never under-shoots, the count-valued analogue of the Bloom filter's deterministic no-false-negatives property; the minimum across d independent rows attenuates the collision noise, since the estimate exceeds the truth only when every row collides with a heavy item.

The error bound

Theorem 10.2.1 (CountMin bound)**.** *With width $w = \lceil e/\epsilon \rceil$ and depth $d = \lceil \ln(1/\delta) \rceil$, for any a the estimate satisfies $f_a \leq \hat{f}_a \leq f_a + \epsilon F$ with probability at least $1 - \delta$.*

Proof. Fix a and a row i. Write the collision mass $X_i = \sum_{b \neq a} f_b \, \mathbf{1}[h_i(b) = h_i(a)] \geq 0$, so $C[i][h_i(a)] = f_a + X_i$. By pairwise independence each b collides with probability $1/w$, so $\mathrm{E}[X_i] = (F - f_a)/w \leq F/w$. Markov's inequality gives $\mathrm{P}(X_i \geq \epsilon F) \leq \mathrm{E}[X_i]/(\epsilon F) \leq 1/(\epsilon w) \leq 1/e$, the last step using $w \geq e/\epsilon$. The d rows use independent hashes, and $\hat{f}_a > f_a + \epsilon F$ only if every row overshoots, so $\mathrm{P}\left(\hat{f}_a > f_a + \epsilon F\right) \leq (1/e)^d \leq \delta$ since $d \geq \ln(1/\delta)$. The lower bound is the deterministic argument above. □

The bound is additive in F, not in f_a: heavy hitters with f_a comparable to F are estimated at small *relative* error, while light items with $f_a \ll \epsilon F$ are washed out. CountMin is the sketch for finding which items are heavy, not for the counts of light ones. The space is $d \cdot w = \lceil \ln(1/\delta) \rceil \lceil e/\epsilon \rceil = O\big((1/\epsilon) \ln(1/\delta)\big)$ counters, logarithmic in the failure probability and inverse in the additive error (Cormode and Muthukrishnan [CM05]).

Remark (Expected overshoot)**.** *The high-probability bound has a tighter expectation behind it. A single row's overshoot has mean $\mathrm{E}[X_i] = (F - f_a)/w$, and at the design width $w = \lceil e/\epsilon \rceil$ this is $\leq F/w = \epsilon F/e$, a factor e below the headline slack ϵF. The factor e is exactly the Markov slack the proof pays to turn a mean into a tail bound; the reported minimum sits below even this per-row mean.* △

The Bernoulli[T] type and merging

CountMin instantiates Chapter 9 with $T = \mathbb{N}$: its channel Q (Definition 9.2.1) has rows indexed by the true count, and the deterministic $\hat{f}_a \geq f_a$ makes Q *upper triangular*, the count-valued analogue of the Bloom filter's one-sided membership matrix. Each row concentrates near the diagonal,

with at most a δ fraction of mass more than ϵF columns to the right. Two sketches built on disjoint substreams with the same parameters and hashes merge by entry-wise addition, $(C_1 \oplus C_2)[i][j] = C_1[i][j] + C_2[i][j]$, which is exactly the sketch of the concatenated stream because UPDATE only increments and addition commutes; the same (ϵ, δ) bound holds at the combined F. Sharing the hash family is what keeps the merge a single correct sketch rather than a doubling of the failure probability, an instance of the Theorem 9.3.1 composition.

Worked example

Take $w = 10$, $d = 5$, so the achieved parameters are $\epsilon = e/w \approx 0.272$ (additive bound $\epsilon F = 0.272\,F$) and $\delta = e^{-d} \approx 0.0067$. On a stream of $F = 1000$ items with a heavy item $f_{a_1} = 410$, the expected per-row overshoot is $(F - f_{a_1})/w = 590/10 = 59$, so each row reports near 469 and the minimum across five rows falls below it; a typical realization returns $\hat{f}_{a_1} = 432$, an overshoot of 22, well inside the bound $\epsilon F = 272$. For a light item with $f = 4$, the expected per-row overshoot is 99.6, and the estimate, though correctly bounded by $4 + 272$, is uninformative about the true count. This is the heavy-hitter design point: heavy items tight relative to their own count, light items only up to the additive slack. Section 10.3 turns to the complementary question of how many *distinct* items a stream contains.

10.3 HyperLogLog

The complementary streaming question is cardinality: estimate $n = |\{x_1, \ldots, x_N\}|$, the number of distinct items, ignoring multiplicity. Exact counting needs $\Theta(n)$ bits in the worst case; HyperLogLog returns an estimate with relative error of order $1/\sqrt{m}$ using m small registers, at space *doubly* logarithmic in n, so a few kilobytes estimate cardinalities into the billions.

The leading-zero trick

Let $h : U \to [0, 1]$ be a uniform hash, read as a binary fraction $0.b_1 b_2 \cdots$, and let $\rho(x)$ be the position of its first 1 bit (leading zeros plus one). For uniform $h(a)$ the bits are fair coins, so $\mathrm{P}\,(\rho \geq k) = 2^{-(k-1)}$, and the maximum of ρ over n distinct items grows as $\log_2 n$ with $O(1)$ spread. Tracking only the running maximum of $\rho(h(a))$ and reporting $2^{\max}$ therefore estimates n from a single small integer, but the multiplicative standard deviation is a factor

of roughly two: an order-of-magnitude check, useless quantitatively. The fix is averaging.

The construction

HyperLogLog averages without a second hash by *stochastic averaging*: the first $\log_2 m$ bits of each item's hash route it to one of m buckets, and the remaining bits feed that bucket's leading-zero count. The m registers act as independent observations of the single-register estimator on streams of expected length n/m.

Definition 10.3.1 (HyperLogLog sketch)**.** *Fix $m = 2^p$ registers ($p \geq 4$) and a hash $h : U \to \{0,1\}^{32}$. The sketch is a register array $M[0..m-1]$, initialized to zero, with* UPDATE(a) *splitting $h(a)$ into a p-bit bucket index j and a remainder w and setting $M[j] \leftarrow \max(M[j], \rho(w))$, and* ESTIMATE$()$ *returning*

$$\hat{n} = \alpha_m \cdot \frac{m^2}{\sum_{j=0}^{m-1} 2^{-M[j]}},$$

with $\alpha_{16} = 0.673$, $\alpha_{32} = 0.697$, $\alpha_{64} = 0.709$, and $\alpha_m = 0.7213/(1+1.079/m)$ for $m \geq 128$.

Algorithm 4: HyperLogLog update and estimate

```
procedure UPDATE(a)
    x ← h(a); j ← first p bits of x; w ← remaining bits;
    M[j] ← max(M[j], ρ(w));
function ESTIMATE()
    Z ← Σ_{j=0}^{m-1} 2^{-M[j]};
    return α_m · m^2 / Z;
```

The sum $Z = \sum_j 2^{-M[j]}$ is the harmonic mean of the per-bucket estimates $2^{M[j]}$ (up to a factor m), which suppresses an outlier register that drew an unlucky run of leading zeros; the m^2 rescales n/m per bucket to n total, and α_m removes the constant bias of the harmonic mean and the discrete ρ, an integral Flajolet et al. [Fla+07] evaluate. For a 32-bit hash with $p = 8$ ($m = 256$), each $M[j] \in \{0, \ldots, 24\}$ fits in 5 bits, so the sketch is $256 \cdot 5 = 1280$ bits (160 bytes) regardless of stream length.

Proposition 10.3.1 (HyperLogLog standard error)**.** *For $m \geq 128$ registers and n large relative to m, the relative standard error of the estimate is $\sigma(\hat{n})/n \approx 1.04/\sqrt{m}$; at $m = 2048$ this is about 2.3%.*

The 1.04 is the asymptotic per-register factor of Flajolet et al. [Fla+07], from an exact generating-function calculation we do not reproduce. The $1/\sqrt{m}$ dependence is the central-limit scaling of m near-independent observations, and doubling the accuracy quadruples the register count. When n is small relative to m (many registers still zero) the algorithm switches to a linear-counting correction $\hat{n} = m \ln(m/V)$ with V the count of zero registers; a saturation correction handles the high-cardinality end.

The Bernoulli[T] type and merging

HyperLogLog instantiates Chapter 9 with $T = \mathbb{N}$, as CountMin does, but its channel Q (Definition 9.2.1) is *not* triangular: it errs both ways. The limiting distribution of $\log \hat{n} - \log n$ is approximately normal with standard deviation $1.04/\sqrt{m}$ [Fla+07], so each row of Q concentrates on a geometric interval around n, symmetric on a multiplicative rather than additive scale, the qualitative contrast with CountMin's one-sided additive error. Two sketches on the same hash and register count merge by entry-wise maximum, $(M^{(1)} \sqcup M^{(2)})[j] = \max(M^{(1)}[j], M^{(2)}[j])$, which is exactly the sketch of the union stream because each bucket stores a maximum and the max over a union is the max of the per-stream maxima; sharing the hash is again the discipline behind the Theorem 9.3.1 composition.

Worked example

Take $m = 256$ and a stream of $n = 10000$ distinct items. The relative standard error from Proposition 10.3.1 is $1.04/\sqrt{256} = 0.065$, so a single sketch typically lands within about ± 650 of 10000, with each register holding the maximum ρ of the $\approx 10000/256 \approx 39$ items in its bucket. The estimate is unbiased but high-variance on one seed: averaging 100 independent seeds returns an empirical mean of 9970 with relative standard deviation 0.063, both matching the proposition. The headline figure is a standard error, not a deterministic bound; the value of the sketch is that the spread shrinks as $1/\sqrt{m}$. Section 10.4 turns to similarity: estimating the Jaccard overlap of two sets from compact signatures.

10.4 MinHash and LSH

The third primitive estimates the *Jaccard similarity* $J(A,B) = |A \cap B|/|A \cup B| \in [0,1]$ of two sets. Exact computation needs space linear in the set sizes, a deal-breaker across a corpus of millions of documents over a vocabulary of millions of shingles. MinHash (Broder [Bro97]) reduces each set to a signature of k integers and estimates J from two signatures in $O(k)$ time.

The collision identity

Fix a hash $h : U \to [0,1]$, uniform and injective on $A \cup B$, and write $\min h(S) = \min_{a \in S} h(a)$.

Proposition 10.4.1 (MinHash collision probability)**.** $\mathrm{P}\left(\min h(A) = \min h(B)\right) = J(A,B)$.

Proof. The minimum of h over $A \cup B$ is attained at a unique a^*. The minima of A and B coincide iff $a^* \in A \cap B$ (otherwise the set not containing a^* has a strictly larger minimum). By uniformity every element of $A \cup B$ is equally likely to be a^*, so $\mathrm{P}\left(a^* \in A \cap B\right) = |A \cap B|/|A \cup B| = J(A,B)$. □

The minimum hash over a union is a uniform sample from the union, and the chance it falls in a subset is the subset's proportion. One hash converts the similarity question into a Bernoulli coin of bias J; estimating J is repetition.

The construction and its error

Definition 10.4.1 (MinHash sketch)**.** *For independent uniform hashes $h_1, \ldots, h_k$, the* signature *of $A \subseteq U$ is $\sigma(A) = (\sigma_1(A), \ldots, \sigma_k(A))$ with $\sigma_i(A) = \min_{a \in A} h_i(a)$, and the* estimator *of $J(A,B)$ is*

$$\hat{J} = \frac{1}{k}\left|\{i : \sigma_i(A) = \sigma_i(B)\}\right|,$$

the fraction of agreeing components.

The signature costs $O(k)$ space and updates in $O(1)$ per inserted item via $\sigma_i \leftarrow \min(\sigma_i, h_i(a))$ (deletion is unsupported, the price of a min-summary).

Proposition 10.4.2 (MinHash error bound)**.** *The estimator satisfies* $\mathrm{E}\left[\hat{J}\right] = J$, $\mathrm{Var}[\hat{J}] = J(1-J)/k$, *and, by Hoeffding's inequality,* $\mathrm{P}\left(|\hat{J} - J| \geq \epsilon\right) \leq 2\exp(-2k\epsilon^2)$ *for every* $\epsilon > 0$.

Algorithm 5: MinHash signature and Jaccard estimate

```
function SIGNATURE(A)
    for i ← 1 to k do
        σ_i ← min_{a∈A} h_i(a);
    end
    return (σ_1, ..., σ_k);
function ESTIMATE(σ(A), σ(B))
    c ← 0;
    for i ← 1 to k do
        if σ_i(A) = σ_i(B) then c ← c + 1;
    end
    return c/k;
```

Proof. Let $Y_i = \mathbf{1}[\sigma_i(A) = \sigma_i(B)]$. By Proposition 10.4.1 each Y_i is Bernoulli(J), independent across i since the h_i are. Then $\hat{J} = \frac{1}{k}\sum Y_i$ has mean J and variance $J(1-J)/k$, and Hoeffding's bound for a mean of k variables in $[0,1]$ gives the tail. □

The error is two-sided and symmetric, like HyperLogLog's (Proposition 10.3.1) and unlike CountMin's. Inverting the tail for a target δ gives accuracy $\epsilon = \sqrt{\ln(2/\delta)/(2k)}$, so the signature length grows linearly in $\ln(1/\delta)$ and quadratically in $1/\epsilon$.

The Bernoulli[T] type

MinHash instantiates Chapter 9 with $T = [0,1]$. Since $k\hat{J}$ is a sum of k independent Bernoulli(J) variables, each row of the channel Q (Definition 9.2.1) is a Binomial(k, J) distribution on $\{0, 1/k, \ldots, 1\}$,

$$Q_{J,\,j/k} = \binom{k}{j} J^j (1-J)^{k-j},$$

symmetric under $\hat{J} \mapsto 1-\hat{J}$, $J \mapsto 1-J$, the contrast with CountMin's triangular and HyperLogLog's log-normal rows. The Proposition 10.4.2 concentration is the channel-matrix view of the Binomial's concentration around its mean.

LSH for approximate nearest neighbour

Estimating J for every pair in a corpus of N documents costs $O(kN^2)$. Locality-sensitive hashing (Indyk and Motwani [IM98]) uses signatures as bucket keys so that similar pairs collide. Partition the length-k signature into b bands of r rows ($k = br$); two sets are *candidates* if they agree on all r rows of at least one band. A band agrees with probability J^r, so the candidate probability is the S-curve

$$p(J) = 1 - (1 - J^r)^b,$$

with a sharp transition near $J^* \approx (1/b)^{1/r}$: larger r steepens and selects, larger b raises and admits. Tuning (b, r) to a target J^* and building the b band tables once turns the $O(N^2)$ comparison into a sub-linear bucket lookup, with an exact pass filtering the small candidate set.

Worked example

Take $|A| = |B| = 100$, $|A \cap B| = 30$, so $|A \cup B| = 170$ and $J = 30/170 \approx 0.176$. With $k = 200$,

$$\mathrm{Var}[\hat{J}] = \frac{0.176 \cdot 0.824}{200} \approx 7.26 \times 10^{-4},$$

a standard error of 0.027. The Hoeffding bound at $\epsilon = 0.05$ gives $2e^{-2 \cdot 200 \cdot 0.0025} = 2e^{-1} \approx 0.74$, loose here because Hoeffding ignores the $J(1 - J)$ factor that shrinks the variance for small J; the exact binomial standard error 0.027 is the relevant figure. A single sketch returns, say, $\hat{J} = 0.185$, within one standard error of 0.176. Halving the error to 0.013 quadruples the signature to $k = 800$, the same $1/\sqrt{k}$ scaling as HyperLogLog. Section 10.5 recaps the common pattern and hands the three sketches to the type algebra of Part III's close.

10.5 Bridge

Three sketches, one skeleton, differing in output type and channel shape. CountMin (Section 10.2) is the frequency sketch: $T = \mathbb{N}$, one-sided, an upper-triangular Q with tail $\mathrm{P}\left(\hat{f}_a \geq f_a + \epsilon F\right) \leq \delta$ (Theorem 10.2.1). HyperLogLog (Section 10.3) is the cardinality sketch: $T = \mathbb{N}$, two-sided and log-normal, with relative error $1.04/\sqrt{m}$ (Proposition 10.3.1) setting the band width. MinHash (Section 10.4) is the similarity sketch: $T = [0, 1]$, binomial rows, Hoeffding-controlled at signature length k (Proposition 10.4.2).

Each comes with a construction, a row-by-row error bound on Q, and a merge rule.

The merge rules share a structure this chapter notes but does not axiomatize. Merging CountMin tables is entry-wise *addition*, HyperLogLog registers entry-wise *maximum*, MinHash signatures (forming the union signature) entry-wise *minimum*; each is associative, commutative, and has an identity (the zero table, the zero register bank, the all-∞ signature), and each respects the Theorem 9.3.1 channel structure. The merge rule is dictated by the algebra of the underlying Bernoulli$[T]$ type, which Chapter 13 treats systematically, showing the three sketches to be instances of the product, sum, and list constructors rather than ad hoc designs. Before that, Chapter 11 turns from streaming summaries to randomized algorithms, where a one-sided Monte Carlo procedure is a Bernoulli$[\{\text{accept}, \text{reject}\}]$ instance and amplification by repetition is composition of one channel with itself, and Chapter 12 lifts the framework from functions to relations for join and multi-attribute queries. The three threads converge in Chapter 13, the close of Part III.

Bibliographic Notes

CountMin. The construction and the additive bound of Theorem 10.2.1 are due to Cormode and Muthukrishnan [CM05]: the $d \times w$ table of pairwise-independent rows, the increment-and-minimum operations, and the $\hat{f}_a \leq f_a + \epsilon F$ guarantee at $w = \lceil e/\epsilon \rceil$, $d = \lceil \ln(1/\delta) \rceil$, proved by the Markov-plus-union-bound argument of Section 10.2. The chapter's addition is the reading of CountMin as a Bernoulli$[\mathbb{N}]$ instance, with the upper-triangular channel encoding the one-sided guarantee and entry-wise merge as composition. Their paper catalogs applications (heavy hitters, quantiles, inner products) the chapter does not cover.

HyperLogLog. The construction and the $1.04/\sqrt{m}$ relative error of Proposition 10.3.1 are due to Flajolet et al. [Fla+07], building on the earlier probabilistic-counting work of Flajolet and Martin. Their generating-function analysis gives the bias correction α_m, the standard-deviation factor, and the limiting log-normal error shape that underwrites the multiplicative band of the Q matrix; the chapter quotes the constants without reproducing the derivation. The max-merge and its equivalence to a sketch of the union are the natural composition under the max-of-leading-zeros structure.

MinHash and LSH. The MinHash sketch is due to Broder [Bro97], whose setting was near-duplicate web detection; the identity $\mathrm{P}\left(\min h(A) = \min h(B)\right) = J(A, B)$ is from that paper, and the k-hash estimator and the Hoeffding bound of Proposition 10.4.2 follow from it. The Binomial-in-k structure of $\hat{J}$, which the chapter reads as the Bernoulli$[[0, 1]]$ channel of Section 10.4, is already implicit there. Locality-sensitive hashing is due to Indyk and Motwani [IM98], for approximate nearest-neighbour search; the MinHash-and-band scheme is the specialization of their general framework to the Jaccard metric, and their paper also covers Hamming and ℓ_p families for readers wanting the general theory.

Chapter 11

Randomized Algorithms

11.1 Miller-Rabin Formal

Chapter 3 §3.4 (Section 3.4) walked through the Miller-Rabin primality test as a Bernoulli[bool] instance, stating the per-round bound $\varepsilon \leq 1/4$ as a fact and applying it. This section closes that deferral, restating the walkthrough as one formal theorem with the per-round bound credited to Rabin and the k-round bound derived from the composition theorem.

Fix an odd $n > 3$, write $n - 1 = 2^r d$ with d odd, and let $W_n(a)$ be the single-round witness test on $a \in \{2, \ldots, n-2\}$, returning $\top$ (PROBABLY-PRIME) or $\bot$ (COMPOSITE), with the predicate "n is prime."

Theorem 11.1.1 (Miller-Rabin as a Bernoulli[bool])**.** *For a drawn uniformly from $\{2, \ldots, n-2\}$, $W_n(a)$ is a* Bernoulli[bool] *instance (Definition 4.5.1) with*

$$\omega = 0 \tag{11.1}$$

for every n, and

$$\varepsilon \leq \tfrac{1}{4} \tag{11.2}$$

for every odd composite $n > 3$. If k independent witnesses are drawn with replacement and the test declares PROBABLY-PRIME *only when every round does, then*

$$\omega^{(k)} = 0 \quad \textit{and} \quad \varepsilon^{(k)} \leq \left(\tfrac{1}{4}\right)^k \tag{11.3}$$

for every odd composite $n > 3$.

Proof. Zero false-negative rate. If n is prime, Fermat's little theorem gives $a^{n-1} \equiv 1$, and in the field $\mathbb{Z}/n\mathbb{Z}$ the only square roots of 1 are ± 1, so

tracing $a^d, a^{2d}, \ldots, a^{2^r d} = 1$ backwards reaches ± 1 and the witness test passes. Every witness yields $\top$, so $\omega = 0$.

Per-round bound. For odd composite n, let $L_n = \{a \in \{2, \ldots, n-2\} : W_n(a) = \top\}$ be the strong-liar set. Rabin's theorem [Rab80], sharpening the conditional analysis of Miller [Mil76], gives $|L_n| \leq (n-3)/4$ via the Chinese remainder structure of $\mathbb{Z}/n\mathbb{Z}$. With a uniform over the pool of size $n-3$, $\varepsilon = |L_n|/(n-3) \leq 1/4$.

The k-round bound. Distinct witnesses are distinct input elements, and sampling with replacement makes the events $\{a_i \in L_n\}$ independent, so Axiom 1 (Axiom 1) applies. The k-round test is the k-fold AND of $W_n(a_1), \ldots, W_n(a_k)$, an FNR-zero chain with $\omega_i = 0$ and $\varepsilon_i \leq 1/4$. By the composition theorem (Theorem 3.3.1) the false-negative rate is $\omega^{(k)} = 1 - \prod_i (1 - \omega_i) = 0$, and the dual all-error direction gives the false-positive rate as the product (a composite survives only if every round is fooled),

$$\varepsilon^{(k)} = \prod_{i=1}^{k} \varepsilon_i \leq \left(\tfrac{1}{4}\right)^k,$$

uniform over odd composite $n > 3$. □

The division of labor is the one chapter 3 promised: number theory supplies the strong-liar bound, the framework supplies the rest. The two error directions take the two faces of the composition theorem, $1 - \prod(1-\omega_i)$ for the monotone false-negative direction and $\prod \varepsilon_i$ for the all-error false-positive direction, exactly as §3.3 distinguishes them. Choosing k to drive $\varepsilon^{(k)}$ below a target δ is the sample-complexity question of Section 11.4.

11.2 Las Vegas vs Monte Carlo

Randomized algorithms divide by where the randomness surfaces: in the runtime, with the output exactly correct, or in the output, with the runtime predictably bounded.

Definition 11.2.1 (Las Vegas algorithm)**.** *A* Las Vegas *algorithm has output that is deterministically correct on every input and coin sequence (when it halts), and a random runtime $T_A(x)$. It is* expected-polynomial-time *when* $\mathrm{E}[T_A(x)]$ *is polynomial in $|x|$.*

Definition 11.2.2 (Monte Carlo algorithm)**.** *A* Monte Carlo *algorithm has runtime deterministically bounded on every coin sequence and output that agrees with the truth only with bounded probability. A one-sided Monte Carlo*

decision algorithm has $\varepsilon \le \alpha$, $\omega = 0$ (or the mirror); a two-sided one has $\varepsilon, \omega \le \alpha$ for some $\alpha < 1/2$.

The two are exhaustive of the useful regimes: an algorithm that is both deterministically correct and worst-case bounded is just deterministic, and one that is neither offers nothing. They are the two faces of one trade-off, since a worst-case runtime bound over all coin sequences leaves no room to keep flipping until the answer is right: if every sequence halts by step $T(|x|)$ and the answer in hand at that step is wrong, the algorithm must return it. Each regime gives up one guarantee. Las Vegas keeps the output exact and pays in the runtime tail; Monte Carlo keeps the runtime fixed and pays in the output. Which to pay on is an application question, exact-but-eventually versus fast-but-occasionally-wrong.

Three examples. *Randomized quicksort* (Las Vegas) sorts by partitioning around a uniform random pivot: the output is always correctly sorted, while the runtime is $O(n \log n)$ in expectation with a $\Theta(n^2)$ tail of probability $n^{-\Omega(c)}$ at $cn \log n$ comparisons. *Karger min-cut* (Monte Carlo) contracts uniform random edges until two super-vertices remain, in deterministic $O(n^2)$ time, returning a fixed min-cut with probability only $\ge 2/[n(n-1)]$; running it r times and keeping the smallest cut fails with probability $\le (1-2/[n(n-1)])^r \le \exp(-2r/[n(n-1)])$, so $r \ge \frac{n(n-1)}{2} \ln(1/\delta)$ trials suffice, a cubic-in-n cost that is the price of a per-trial signal as weak as $1/n^2$. *Miller-Rabin* (Monte Carlo, Section 11.1) runs in $O(k \log^3 n)$ and errs only on composites, at rate $\le (1/4)^k$ by Theorem 11.1.1; its per-round signal $1/4$ is independent of input size, so amplification needs only $O(\log(1/\delta))$ rounds, the opposite extreme from Karger.

Conversion. The regimes interconvert at a cost. A Las Vegas algorithm with expected runtime $\mu(|x|)$ becomes Monte Carlo by running for $c\,\mu(|x|)$ steps and reporting failure on timeout: by Markov the failure probability is $\le 1/c$, so $c = 10$ gives error ≤ 0.1 at worst-case runtime $10\mu(|x|)$. This is why expected polynomial time is essentially worst-case polynomial time with bounded error. The reverse needs a verifier: a Monte Carlo algorithm with a deterministic polynomial-time check of candidate outputs becomes Las Vegas by repeating until a verified result appears, in expected $1/(1-\varepsilon)$ trials. The verifier is a nontrivial assumption, which is why ZPP = RP $\cap$ coRP (Section 11.3) but the directions are not symmetric.

The Bernoulli[bool] **correspondence.** A Monte Carlo *decision* algorithm returns a Boolean $A(x)$ guessing a predicate $P(x)$, with error structure exactly the 2×2 channel of Definition 4.5.1, $\varepsilon = \mathrm{P}\left(A(x) = \top \mid P(x) = \bot\right)$ and $\omega = \mathrm{P}\left(A(x) = \bot \mid P(x) = \top\right)$. It *is* a Bernoulli[bool] instance, the k-fold amplification of Theorem 11.1.1 is the composition theorem (Theorem 3.3.1) on independent instances, and chapter 7's PPV and NPV apply with prevalence the base rate of true positives. A Las Vegas algorithm has no error to rate, its output being deterministically correct and its randomness living in the runtime; its analysis belongs to runtime tail bounds (Markov, Chebyshev, Chernoff of Section 11.4), not to the Bernoulli channel. From here, "randomized algorithm" means a Monte Carlo decision algorithm, read as a Bernoulli[bool] instance.

11.3 RP, BPP, and Bernoulli Type

The Monte Carlo decision problems (Definition 11.2.2) carve into the classes RP, coRP, ZPP, and BPP by where the error is allowed to land. Each is a location on the chapter-6 rate plane: RP and coRP on the two axes of the perfect filter line, ZPP at the origin, BPP in the interior. The correspondence is not an analogy; the complexity-class definitions and the chapter-6 classification are the same axes from two angles.

RP: certain accept, the FPR-zero axis

Definition 11.3.1 (Randomized Polynomial Time, RP)**.** *A language L is in* RP *when a polynomial-time randomized A satisfies $x \in L \Rightarrow \mathrm{P}\left(A(x) = 1\right) \geq 1/2$ and $x \notin L \Rightarrow \mathrm{P}\left(A(x) = 1\right) = 0$, the probability over A's coins.*

Read $A(x) = 1$ as the report IN-L. The second line says the algorithm never accepts a non-member, $\varepsilon = \mathrm{P}\left(A(x) = 1 \mid x \notin L\right) = 0$; the first says it accepts a member with probability $\geq 1/2$, $\omega = \mathrm{P}\left(A(x) = 0 \mid x \in L\right) \leq 1/2$. An RP algorithm is a Bernoulli[bool] instance with $\varepsilon = 0$, $\omega \leq 1/2$: it sits on the *FPR-zero* axis of the perfect filter line (Definition 6.4.1). An accept is certain, a witness seen is a witness believed, and the only failure is missing a witness that exists. This is the certain-accept shape, the decision-algorithm kin of the perfect hash filter of Chapter 15, not the Bloom filter, whose certainty is on the reject side.

The canonical RP problem is NON-ZERO POLYNOMIAL: given a circuit for a polynomial f over a finite field, decide whether $f \not\equiv 0$. By Schwartz–Zippel, a random assignment makes a non-zero f evaluate non-zero with

probability $\geq 1 - d/|F|$ (degree d, field size $|F|$), while a zero f always evaluates zero. Reporting NON-ZERO on a non-zero evaluation is the RP shape: silent on zero polynomials ($\varepsilon = 0$), occasionally missing a non-zero one ($\omega \leq d/|F|$).

coRP and ZPP: the mirror axis and the origin

Definition 11.3.2 (coRP and ZPP)**.** $L \in \text{coRP}$ *when* $\bar{L} \in \text{RP}$*: equivalently* $x \in L \Rightarrow \text{P}(A(x) = 1) = 1$ *and* $x \notin L \Rightarrow \text{P}(A(x) = 1) \leq 1/2$. $L \in \text{ZPP}$ *when* $L \in \text{RP} \cap \text{coRP}$.

A coRP algorithm has $\omega = 0$, $\varepsilon \leq 1/2$: it sits on the *FNR-zero* axis, the certain-*reject* shape, the decision-algorithm kin of the Bloom filter (Definition 5.2.1), whose $\perp$ report is certain. Miller-Rabin (Theorem 11.1.1) is the canonical example: a composite witness certifies compositeness, so on the language PRIMES a reject is certain ($\omega = 0$, a prime is never rejected) while a composite is occasionally accepted as prime ($\varepsilon \leq 1/4$). Thus PRIMES $\in$ coRP, and dually COMPOSITES $\in$ RP via the same algorithm, where instead a composite-witness is a certain *accept* of compositeness. (This was the textbook RP/coRP example until the deterministic AKS test of 2002.) ZPP, at $\varepsilon = \omega = 0$, is the origin: exact decisions paid for in the runtime tail, equivalently the expected-polynomial-time Las Vegas class.

BPP: the interior

Definition 11.3.3 (Bounded-error Probabilistic Polynomial Time, BPP)**.** $L \in \text{BPP}$ *when a polynomial-time randomized* A *satisfies* $x \in L \Rightarrow \text{P}(A(x) = 1) \geq 2/3$ *and* $x \notin L \Rightarrow \text{P}(A(x) = 1) \leq 1/3$.

Read as a channel, $\omega, \varepsilon \leq 1/3$: BPP sits in the interior of the rate plane, off both axes. The chapter-7 vocabulary applies directly. At prevalence $\pi = \text{P}(x \in L)$ the positive predictive value (Definition 7.4.1) is

$$\text{PPV} = \frac{\pi(1-\omega)}{\pi(1-\omega) + (1-\pi)\varepsilon} \geq \frac{\pi \cdot 2/3}{\pi \cdot 2/3 + (1-\pi) \cdot 1/3} = \frac{2\pi}{\pi + 1},$$

which at $\pi = 0.01$ is only ≈ 0.0198: despite 2/3-versus-1/3 per-class accuracy, 98% of accept reports are false at low prevalence, the same prevalence-dependence Definition 7.4.2 and chapter 7 record. The 2/3-versus- 1/3 gap is conventional but load-bearing: it forces a signal in both directions so amplification can drive both errors down, where the gap-closed class PP cannot.

Amplification

Theorem 11.3.1 (RP amplification)**.** *For an RP algorithm with $\omega \leq 1/2$, running k independent times and accepting if any run accepts gives an RP algorithm with $\omega \leq 2^{-k}$, $\varepsilon = 0$.*

This is the composition theorem (Theorem 3.3.1) with the OR aggregator: the certain-accept property ($\varepsilon = 0$) is preserved (any accept is true), and ω at the OR is the product of per-run rates by independence (Axiom 1). It is the FPR-zero mirror of the k-round Miller-Rabin bound, which drives the coRP rate ε down via AND; coRP amplifies dually. Bringing ω to 2^{-30} costs 30 runs.

Theorem 11.3.2 (BPP amplification)**.** *For a BPP algorithm with $\omega, \varepsilon \leq 1/3$, the majority of k independent runs gives $\omega, \varepsilon \leq \exp(-\Omega(k))$.*

The exponent is no longer the per-run rate but the Chernoff exponent for a majority of independent draws; the proof, via the Chernoff bound (Theorem 11.4.1), is in Section 11.4. BPP error decays exponentially in the repetition count even though no single coin sequence drives either error to zero.

Remark (The quantum sibling: amplitude amplification)**.** *The amplification theorems above compose* probabilities *across independent runs: the OR aggregator drives ω to 2^{-k} (Theorem 11.3.1), the majority aggregator gives Chernoff decay (Theorem 11.3.2). The quantum class BQP has a sibling operation,* amplitude amplification *(Grover [Gro96] search is its special case; the general form is [Bra+02]), that composes* amplitudes *instead, and the comparison is exact. Take a one-sided routine that succeeds with probability p. Classical OR-amplification reaches near-certainty in $k \propto 1/p$ runs, since the found-probability $1-(1-p)^k \approx kp$ grows linearly in k. Amplitude amplification rotates the success* amplitude *by a fixed angle θ per step, $\sin\theta = \sqrt{p}$, so after k steps the success probability is $\sin^2\big((2k+1)\theta\big)$, the square of an amplitude that grows linearly in k, reaching near-certainty in $k \propto 1/\sqrt{p}$ steps. The quadratic speedup is exactly probability equals amplitude squared, applied to a quantity that accumulates linearly: the Born-rule square is the entire $\sqrt{\cdot}$.*

Two caveats keep this an analogy rather than an identity, and both trace to the latent-value question of Section 16.5. First, the classical theorems presuppose a stable latent value to resample: Axiom 1 makes the repeated looks independent draws on one fixed truth. A superposition has no such value; measuring collapses it, and re-measuring in the same basis returns

the collapsed outcome, not a fresh sample, so "repeat and aggregate" is unavailable and one must amplify the amplitude coherently before the single measurement. Second, classical amplification is monotone, more runs never hurt and the error tends to zero, whereas amplitude amplification is oscillatory: rotating past $(2k+1)\theta \approx \pi/2$ *drives the success probability back down. The same Born square that buys the speedup forbids the monotonicity.* △

The correspondence at scale

Class	ε bound	ω bound
RP	$= 0$	$\leq 1/2$
coRP	$\leq 1/2$	$= 0$
ZPP	$= 0$	$= 0$
BPP	$\leq 1/3$	$\leq 1/3$

The first three rows trace the perfect filter line of Definition 6.4.1: RP the FPR-zero axis, coRP the FNR-zero axis, ZPP their intersection at the origin. BPP is the interior below 1/3. The chapter-6 classification of approximate sets by axis and the classical randomized-complexity taxonomy are one classification applied to two families, arbitrary Bernoulli[bool] instances there, polynomially constrained decision algorithms here. The one-sided-versus-two-sided distinction of Section 5.4 is what separates RP and coRP from BPP, and amplification is the constraint that propagates: any instance with per-trial error bounded away from 1/2 on both inputs is driven to negligible error by repetition and the right aggregator (OR or AND on an axis, majority in the interior). Chapters 12 and 13 lift these classes along with the rest of the framework, so the relational analog of BPP amplifies for the same reason.

11.4 Sample Complexity

Every algorithm of this chapter has a knob k, the round or repetition count, that drives a per-trial error below a target δ. The sample-complexity question is how to set it, and the answer rests on one theorem about sums of independent Bernoulli variables.

Theorem 11.4.1 (Multiplicative Chernoff bound)**.** *For independent* $X_1, \ldots, X_n$ *with* $\mathrm{P}(X_i = 1) = p$*,* $X = \sum_i X_i$*,* $\mu = np$*, and every* $\epsilon \in (0, 1]$*,*

$$\mathrm{P}(X \geq (1+\epsilon)\mu) \leq \exp\left(-\tfrac{\mu\epsilon^2}{3}\right), \qquad \mathrm{P}(X \leq (1-\epsilon)\mu) \leq \exp\left(-\tfrac{\mu\epsilon^2}{2}\right).$$

Proof. For $t > 0$, Markov on e^{tX} gives $\mathrm{P}(X \geq a) \leq e^{-ta}\mathrm{E}\left[e^{tX}\right]$, and independence factors the moment-generating function, $\mathrm{E}\left[e^{tX}\right] = \prod_i (1 - p + pe^t) = (1 + p(e^t - 1))^n \leq \exp(\mu(e^t - 1))$ by $1 + x \leq e^x$. At $a = (1+\epsilon)\mu$ the exponent $-t(1+\epsilon)\mu + \mu(e^t - 1)$ is minimized at $t = \ln(1+\epsilon)$, giving $-\mu[(1+\epsilon)\ln(1+\epsilon) - \epsilon]$, and $(1+\epsilon)\ln(1+\epsilon) - \epsilon \geq \epsilon^2/3$ on $(0, 1]$ yields the upper tail. The lower tail follows with $t < 0$ and the matching $\geq \epsilon^2/2$ estimate. □

The upper tail's $\mu\epsilon^2/3$ is looser than the lower tail's $\mu\epsilon^2/2$ (deviations above the mean are harder against the $\{0, 1\}$ ceiling); a two-sided union gives $\mathrm{P}(|X - \mu| \geq \epsilon\mu) \leq 2\exp(-\mu\epsilon^2/3)$.

Miller-Rabin round count. Miller-Rabin sits on the FNR-zero axis ($\omega = 0$, $\varepsilon \leq 1/4$), so the Chernoff bound is not needed: a single fooled round suffices for a false positive, and the composition theorem reduces $\varepsilon^{(k)}$ to the product $\leq (1/4)^k$ (Theorem 11.1.1). Driving this below δ requires $k \geq \log_4(1/\delta) = \frac{1}{2}\log_2(1/\delta)$: about 10 rounds for $\delta = 10^{-6}$, 20 for 10^{-12}, 64 for 2^{-128}. Each bit of confidence costs a constant number of rounds, not a constant factor, because the error is one-sided and a single witness collapses the chain.

BPP amplification. The deferred proof of Theorem 11.3.2 is a Chernoff calculation. Let A accept correctly with probability $\geq 1/2 + \epsilon_0$ ($\epsilon_0 = 1/6$ for the 2/3-versus-1/3 gap of Definition 11.3.3), let $X_i = 1$ when run i is correct, independent by Axiom 1, and let $X = \sum_{i=1}^{k} X_i$ with $\mu = k(1/2 + \epsilon_0)$. The majority is wrong when $X < k/2 = (1 - \epsilon')\mu$ with $\epsilon' = \epsilon_0/(1/2 + \epsilon_0)$, so the lower tail gives

$$\mathrm{P}(X < k/2) \leq \exp\!\left(-\tfrac{\mu(\epsilon')^2}{2}\right) = \exp\!\left(-\tfrac{k\epsilon_0^2}{2(1/2+\epsilon_0)}\right) \leq \exp\!\left(-\tfrac{k\epsilon_0^2}{3}\right),$$

the last step since $1/[2(1/2 + \epsilon_0)] \geq 1/3$ for $\epsilon_0 \leq 1$. Asking this be $\leq \delta$ gives $k \geq 3\ln(1/\delta)/\epsilon_0^2$, so the $\epsilon_0 = 1/6$ gap costs $108\ln(1/\delta)$ rounds (the constant tightens with the sharper lower-tail exponent). The point is the shape: BPP amplification is $O((1/\epsilon_0^2)\log(1/\delta))$, exponential decay in k with the exponent the squared margin above 1/2. Against RP amplification's bit-per-round (ω from 1/2 to 2^{-k} in k rounds), the two-sided contract pays a $\Theta(1/\epsilon_0^2)$ overhead per bit, the price of needing a vote rather than a single witness.

Chernoff versus the variance interval. The Chernoff bound and the variance interval of Section 7.2 are two tails of the same binomial estimator $\hat{p} = X/n$. Chebyshev on the variance $p(1-p)/n$ (Proposition 7.2.2) gives $\mathrm{P}(|\hat{p} - p| \geq \epsilon) \leq p(1-p)/(n\epsilon^2) \leq 1/(4n\epsilon^2)$, polynomial in $n\epsilon^2$; Hoeffding, the additive cousin of Theorem 11.4.1, gives $\leq 2\exp(-2n\epsilon^2)$, exponential. Chebyshev needs only the variance and gives the chapter-7 point estimate its Cramér–Rao certificate; Chernoff needs the full moment-generating function and gives sample complexity. The gap between the polynomial $n = \Theta(1/\delta)$ and the exponential $n = \Theta(\log(1/\delta))$ is the gap between 10^6 samples for $\delta = 10^{-6}$ and 20, which is why Chernoff, not Chebyshev, is the workhorse of randomized-algorithm analysis.

11.5 Bridge

The chapter traced one figure from four angles. Section 11.1 closed the chapter-3 deferral, making Miller-Rabin a formal Bernoulli[bool] with $\omega = 0$ and $\varepsilon \leq 1/4$ per round (Theorem 11.1.1). Section 11.2 separated the Monte Carlo regime, where Bernoulli[bool] analysis applies, from its exact-output Las Vegas sibling. Section 11.3 read the Monte Carlo classes as locations on the rate plane of Definition 6.4.1: RP (Definition 11.3.1) on the *FPR-zero* axis (certain accept), coRP on the *FNR-zero* axis (certain reject, the Bloom-filter shape), ZPP at the origin, BPP in the interior, with the one-sided-versus-two-sided split of Section 5.4 separating RP and coRP from BPP. Section 11.4 quantified the repetition count, the Chernoff bound (Theorem 11.4.1) giving the $k = \Theta(\log(1/\delta))$ that makes randomized algorithms practical and sharpening the chapter-7 variance interval (Section 7.2) from polynomial to exponential decay.

The picture extends in two directions before Part III closes. Chapter 12 lifts the analysis to Bernoulli[R] for binary relations, where the question is whether x and y stand in a relation rather than whether x belongs to a set, the setting for joins and pair predicates; a Monte Carlo relation test is a Bernoulli[R] instance with a per-pair rate, and amplification carries over unchanged. Chapter 13 closes Part III by giving the type algebra under which the chapter-10 sketches, this chapter's randomized algorithms, and the chapter-12 relations are all instances of a few constructors over Bernoulli[T], composing by structure.

Bibliographic Notes

Miller-Rabin. The per-round bound $\varepsilon \leq 1/4$ of Theorem 11.1.1 is Rabin [Rab80]'s strong-liar theorem, which replaced the generalized Riemann hypothesis that Miller [Mil76]'s deterministic test assumed with random base sampling, trading determinism for an unconditional $1/4$ rather than $1/2$ guarantee. The number theory is theirs; the framework reading, with the witness predicate a Bernoulli[bool] and $\varepsilon^{(k)} \leq (1/4)^k$ a corollary of the chapter-3 composition theorem, is the chapter's. A comparison of the Miller and Rabin tests is in Cormen et al. [Cor+09, Ch. 31]; the AKS test (2002) supersedes both asymptotically but loses to k-round Miller-Rabin in the constant factors of practical key generation, which is why Miller-Rabin remains the standard library implementation.

Randomized algorithms and concentration. The standard graduate reference is Motwani and Raghavan [MR95], covering the Las Vegas / Monte Carlo distinction of Section 11.2, the RP/BPP/ZPP classification of Section 11.3, and the tail bounds of Section 11.4, organized by technique where this chapter organizes by error structure. The multiplicative Chernoff bound of Theorem 11.4.1 originates in Chernoff's 1952 paper; the formulation and the BPP amplification follow Mitzenmacher and Upfal [MU17, Ch. 4], which also develops the additive Hoeffding form that underwrote the chapter-10 CountMin and HyperLogLog bounds. The choice between additive and multiplicative forms is problem-driven.

Karger min-cut. The contraction algorithm of Section 11.2 is due to Karger [Kar93]: a fixed min-cut survives all contractions with probability $\geq 2/[n(n-1)]$, so $\Theta(n^2 \log n)$ trials find it with high probability. Its value here is not its complexity but its illustration of the Monte Carlo pattern, a weak per-trial signal amplified by independent repetition, the same Bernoulli[bool] amplification Section 11.4 applies to Miller-Rabin and BPP.

Quantum amplitude amplification. The quantum sibling of the amplification theorems, noted in Section 11.3, is amplitude amplification. Grover [Gro96], "A Fast Quantum Mechanical Algorithm for Database Search," gave the search whose state rotates toward the marked item by a fixed angle per iteration; Brassard et al. [Bra+02], "Quantum Amplitude Amplification and Estimation," generalized it to any one-sided routine and is the reference for the $1/\sqrt{p}$ scaling. The reading here, that the quadratic speedup

over classical OR-amplification is the Born-rule square applied to a linearly accumulating amplitude, and that it needs coherence in place of the resampleable latent value the classical theorems assume (Section 16.5), is the chapter's.

Chapter 12

Approximate Relations

12.1 From Approximate Sets to Approximate Relations

A binary relation $R \subseteq T \times U$ is a set whose elements are ordered pairs, so the chapter-5 construction (Chapter 5) applies with the universe $T \times U$ in place of U. A relation is its membership predicate $\mathbb{1}_R : T \times U \to \{\top, \bot\}$ (Section 5.1), and an *approximate relation* is the noisy analogue,

$$\tilde{R} : T \times U \longrightarrow \text{Bernoulli[bool]},$$

whose evaluation $\tilde{R}(t, u)$ reports whether $(t, u) \in R$ with the two rates of Definition 4.5.1,

$$\varepsilon = \mathrm{P}\left(\tilde{R}(t,u) = \top | (t,u) \notin R\right), \qquad \omega = \mathrm{P}\left(\tilde{R}(t,u) = \bot | (t,u) \in R\right).$$

The block structure is two-block, the in-relation block R and its complement, one rate per block. The universe grew; the apparatus did not.

Two examples fix the picture. The *friend-of* relation $F \subseteq P \times P$ on a population P stores known friend pairs in a Bloom-style structure: a query $\tilde{F}(\text{Alice}, \text{Bob})$ reads the hashed positions for the pair, with $\omega = 0$ (a stored pair has its bits set) and a tunable ε, the chapter-5 Bloom analysis applied to pairs. The *word-in-document* relation $\tilde{R} \subseteq W \times D$, the inverted index of information retrieval, holds (w, d) when word w appears in document d, representable as a Bloom filter per document; the chapter-5 space-versus-accuracy trade carries over verbatim.

The chapter-9 view (Chapter 9, Section 9.2) is equivalent: $\mathbb{1}_R$ is a binary function $T \times U \to$ bool whose channel has one row per input pair, collapsing under Axiom 2 to model order two (in the relation or not), exactly as

chapter 5 collapsed the per-element rates of an approximate set. Whether to read $\tilde{R}$ as an approximate set on $T \times U$ or as an approximate binary function is a matter of which downstream machinery is nearer; both are the same object. Section 12.2 adds structure on the universe itself, the relational operators of join, projection, and selection, with rates propagating by the chapter-6 composition theorems lifted to pairs; Section 12.4 then recovers the MinHash similarity join of chapter 10 as one instance.

12.2 Composition of Approximate Relations

The universe $T \times U$ carries structure beyond "set of pairs": a relation admits projection, selection, and join. Each operation lifts to approximate relations, and the propagation of (ε, ω) follows from chapter 3's AND/OR analysis (Section 3.1) and chapter 6's intersection theorem (Proposition 6.2.1) with pairs in place of elements.

Natural join

The natural join of $R_1 \subseteq T \times U$ and $R_2 \subseteq U \times V$ on the shared attribute is $R_1 \bowtie R_2 = \{(t, u, v) : (t, u) \in R_1,\ (u, v) \in R_2\}$, with membership predicate $\mathbb{1}_{R_1}(t, u) \wedge \mathbb{1}_{R_2}(u, v)$. The approximate join reads as the AND of two noisy Booleans, $(\tilde{R}_1 \bowtie \tilde{R}_2)(t, u, v) = \tilde{R}_1(t, u) \wedge \tilde{R}_2(u, v)$, the object whose rates chapter 3 derived, with one added hypothesis: cross-relation independence, that the per-query randomness of $\tilde{R}_1$ at (t, u) is independent of that of $\tilde{R}_2$ at (u, v). This is Axiom 1 applied across the two structures, holding when they are built and queried independently.

Theorem 12.2.1 (Join error rates)**.** *Let $\tilde{R}_1, \tilde{R}_2$ be FNR-zero approximate relations with false-positive rates $\varepsilon_1, \varepsilon_2$, under cross-relation independence. Then $\tilde{R}_1 \bowtie \tilde{R}_2$ has $\omega_{\bowtie} = 0$ and, over the three regions of out-of-join triples (the pair matched in R_1 but not R_2, in R_2 but not R_1, in neither) with weights w_1, w_2, w_3,*

$$\varepsilon_{\bowtie} \;=\; w_1\varepsilon_2 + w_2\varepsilon_1 + w_3\varepsilon_1\varepsilon_2.$$

Proof. A true join triple has both component pairs in their relations, each reported $\top$ with probability $1 - \omega_i = 1$, so $\omega_{\bowtie} = 0$. A false positive needs both components to report $\top$ at an out-of-join triple, and the per-region probability depends on which components are genuinely present: on the region where $(t, u) \in R_1$ but $(u, v) \notin R_2$, the first reports $\top$ correctly while the second false-positives, $(1 - \omega_1)\varepsilon_2 = \varepsilon_2$; on the mirror region, ε_1; on the

both-absent region, $\varepsilon_1\varepsilon_2$. Total probability over the three regions gives the stated rate. This is Proposition 6.2.1 of chapter 6 on the triple universe, with the in-join block as $A \cap B$. □

The bare product $\varepsilon_1\varepsilon_2$ is only the both-absent term: it is exact when out-of-join triples are dominated by the region where *neither* component pair is present, which holds when both relations are sparse in a large triple universe, the typical database regime. Otherwise the linear terms $w_1\varepsilon_2 + w_2\varepsilon_1$ dominate, because a triple with one component genuinely present false-positives at the surviving single rate, not at the product. If either relation has $\omega_i > 0$, the join inherits a positive $\omega_{\bowtie}$ by the full intersection formula; the FNR-zero case is the one-sided special case.

A join breaks element-wise independence

Theorem 12.2.1 gives the rate of *one* output triple. It is silent on a sharper question: are the output triples independent of one another? They are not, and the failure is the one place the lift to relations breaks Axiom 1 (Axiom 1) rather than merely re-deriving its consequences.

Theorem 12.2.2 (A join does not preserve element-wise independence)**.** *The approximate join $\tilde{R}_1 \bowtie \tilde{R}_2$ is a Bernoulli relation only* marginally*. Each output triple carries the rates of Theorem 12.2.1, but the joint law over output triples does not factor: two output triples that share a component pair are positively correlated, so element-wise independence (Axiom 1) fails for the join.*

Proof. Take two output triples (t, u, v_1) and (t, u, v_2) with $v_1 \neq v_2$ that share the same R_1-pair (t, u). Both acceptance events contain the *same* trial $A = \{\tilde{R}_1(t, u) = \top\}$ as a conjunct:

$$\{(t, u, v_i) \in \tilde{R}_1 \bowtie \tilde{R}_2\} = A \cap B_i, \qquad B_i = \{\tilde{R}_2(u, v_i) = \top\},$$

where A, B_1, B_2 are independent under cross-relation independence. Write $a = \mathrm{P}(A)$. The two triples are accepted together with probability

$$\mathrm{P}(A \cap B_1 \cap B_2) = a\,\mathrm{P}(B_1)\,\mathrm{P}(B_2),$$

whereas the product of their marginals is $(a\mathrm{P}(B_1))(a\mathrm{P}(B_2)) = a^2\mathrm{P}(B_1)\,\mathrm{P}(B_2)$. For $0 < a < 1$ the two differ by the factor $1/a > 1$, so the joint acceptance exceeds the independent-model value and the triples are positively correlated. The shared trial $\tilde{R}_1(t, u)$ is reused by every output triple built from (t, u), welding those triples together; the same holds for triples sharing an R_2-pair. □

A concrete reading. Suppose the shared R_1-pair fires with probability $a = 0.9$ and each of $\tilde{R}_2(u, v_1)$, $\tilde{R}_2(u, v_2)$ fires with probability 0.9. The two triples are accepted together with probability $0.9 \cdot 0.9 \cdot 0.9 = 0.729$, whereas an independence assumption would predict $0.81 \cdot 0.81 = 0.6561$; the truth exceeds it by the factor $1/0.9 \approx 1.11$. A query planner that estimated the chance both rows survive as a product of per-row probabilities would understate it.

The marginal rates of Theorem 12.2.1 remain exactly right for any *single* triple, and an expected false-positive count over a join still adds tuple by tuple, since linearity of expectation needs no independence. It is the higher moments, and any all-correct probability computed as a product over output tuples, that feel the correlation. The next two operators do preserve independence, which is exactly what makes the join the exception.

Projection and selection

The projection $\pi_T(R) = \{t : \exists u, (t, u) \in R\}$ collapses the U axis by an existential quantifier, so $t \in \pi_T(\tilde{R})$ iff some pair in the *fiber* of t, its preimage $\{(t, u) : u \in U\}$, reports $\top$. Projection is a logical OR over the fiber, and its *fan-in* is the fiber size $k(t)$. An FNR-zero input gives an FNR-zero projection (a true t has a witness pair reporting $\top$), but the FPR *amplifies*: a non-projecting t has all $k(t)$ fiber pairs outside the relation, each false-positive at rate ε independently, so

$$\varepsilon_\pi(t) = 1 - (1 - \varepsilon)^{k(t)} \;\geq\; \varepsilon,$$

climbing toward 1 with the fan-in. The true-positive side is boosted in step: a non-member fiber pair can itself false-positive and so rescue the projection of a genuine t, giving $\tau_\pi(t) = 1 - (1 - \tau)^j (1 - \varepsilon)^{k(t)-j}$ when $j \geq 1$ of the fiber pairs are genuine. For a binary relation projected onto T the fiber is the whole U-axis, so $k(t) = |U|$ is uniform and the rate is $1 - (1 - \varepsilon)^{|U|}$; in a relation of higher arity the fan-in varies from one projected tuple to the next.

That variation is structural. Distinct projected elements have *disjoint* fibers, so they read disjoint sets of trials and stay mutually independent: projection preserves Axiom 1 (Axiom 1), unlike the join. But the per-element rate $1 - (1 - \varepsilon)^{k(t)}$ depends on the fan-in $k(t)$, which need not be constant, so $\pi_T(\tilde{R})$ is in general a higher-order Bernoulli relation, one rate block per distinct fan-in value, collapsing to a single first-order rate only when the fan-in is uniform. Existential quantification over a large co-domain

inflates any false-positive bias, the one structural penalty the lift to relations adds, and the motivation for the sketch-based projection of Section 12.4.

The selection $\sigma_P(R) = \{(t, u) \in R : P(t, u) = \top\}$ passes the rates through unchanged when P is exact (a deterministic restriction of the support). When P is itself approximate with rate ε_P, the selection predicate is the AND $\tilde{R}(t, u) \wedge \tilde{P}(t, u)$, evaluated at the same pair, so its FPR is the three-region rate of Theorem 12.2.1 again, $\approx \varepsilon\varepsilon_P$ only in the both-absent regime.

Worked example

Let $\tilde{F}$ be a Bloom-style friend-of relation with $\omega_F = 0$, $\varepsilon_F = 0.05$, and $\tilde{C}$ a coworker relation with $\omega_C = 0$, $\varepsilon_C = 0.02$, built and queried independently. The join $\tilde{F} \bowtie \tilde{C}$ returns triples (p_1, p_2, p_3) where $\tilde{F}$ reports p_1 a friend of p_2 and $\tilde{C}$ reports p_2 a coworker of p_3. By Theorem 12.2.1, $\omega_{\bowtie} = 0$ and $\varepsilon_{\bowtie} = w_1\varepsilon_C + w_2\varepsilon_F + w_3\varepsilon_F\varepsilon_C$. In a sparse population, where almost every out-of-join triple has p_1 unrelated to p_2 *and* p_2 unrelated to p_3, the both-absent term dominates and $\varepsilon_{\bowtie} \approx \varepsilon_F\varepsilon_C = 0.001$, three orders below either operand, the relational face of the chapter-6 intersection gain. But the gain is conditional: a triple where p_1 truly is a friend of p_2 while p_2 is not a coworker of p_3 false-positives at $\varepsilon_C = 0.02$, the linear rate, so where such half-matched triples are common the join rate sits closer to $\max(\varepsilon_F, \varepsilon_C)$ than to the product. The cost is in the assumptions, not the algebra: cross-relation independence must hold (shared hash families break the product), and the relations must be sparse enough for the both-absent region to dominate.

Composing the operators

The three operators compose in any order, with the *marginal* rate of any single output tuple propagating by induction on the expression tree. A query $\pi_T(\sigma_P(\tilde{R}_1 \bowtie \tilde{R}_2))$ evaluates bottom-up: the join gives Theorem 12.2.1's rate, the exact selection passes it through, the projection inflates the false-positive side by the fan-in factor and preserves the structural zero. Each operator has a closed-form rate-update rule for a single tuple, and chaining them gives the per-tuple rule for the whole expression, the same closure-under-composition chapter 6 told for sets.

Two cautions separate the relational case from the set case. The projection inflation is one: a union is a single Boolean OR per query with no existential quantifier over a domain of unknown size, while a projection

quantifies over the full second attribute, and Section 12.4 replaces it with a sketch that estimates the quantifier without paying the fan-in amplification in full. The join correlation is the other and the deeper one. The bottom-up rates above are marginal, exact for any single output tuple, but Theorem 12.2.2 showed that a join correlates the tuples it produces, so the probability that the *whole* result is error-free does not factor as a product of per-tuple rates. Composing marginal rates is sound; composing them as though the output tuples were independent is not, once a join sits anywhere in the tree.

12.3 Preservation of Relational Invariants

A relation often carries a structural invariant: reflexivity, symmetry, single-valuedness, transitivity. An exact relation either has the property or it does not. An approximate relation tests each pair through its own Bernoulli trial (Axiom 1), and a property that constrains *many* pairs at once survives only when all of the relevant trials cooperate. The lesson of this section is one of fragility: the probability that an approximate relation inherits a global invariant is a product over the constrained pairs, and it tends to zero as the ground universe grows. Where a cheap repair exists, as for symmetry, we exhibit it and pay its rate cost; where none does, as for transitivity, we state the honest limiting behavior rather than force a clean formula.

Throughout, $R \subseteq A \times A$ is an exact relation with the named property and $\tilde{R}$ its approximation, with false-positive rate ε, true-positive rate $\tau = 1 - \omega$, and the per-pair trials independent.

Reflexivity

A relation is *reflexive* when it contains the whole diagonal: $(a, a) \in R$ for every $a \in A$. The approximation is reflexive only if each of the $|A|$ diagonal pairs survives as a true positive.

Theorem 12.3.1 (Reflexivity is preserved with vanishing probability)**.** *If R is reflexive, then $\tilde{R}$ is reflexive with probability $\tau^{|A|}$, which tends to* 0 *as $|A| \to \infty$ for any $\tau < 1$.*

Proof. Each diagonal pair (a, a) is a true positive with probability τ, and the $|A|$ diagonal trials are independent (Axiom 1). Reflexivity holds exactly when all of them fire, with probability $\tau^{|A|}$, which decays geometrically in $|A|$ since $\tau < 1$. □

Symmetry and its closure

A relation is *symmetric* when $(a, b) \in R$ implies $(b, a) \in R$. The two ordered trials of an unordered pair are tested independently and need not agree, so symmetry, like reflexivity, is preserved only with a probability that vanishes as the number of pairs grows. Unlike reflexivity, symmetry admits a cheap repair: rather than hope the two ordered trials agree, force agreement by taking the *symmetric closure*, which admits a pair when *either* ordered trial fires.

Theorem 12.3.2 (Symmetric closure)**.** *Let R be symmetric and $\tilde{R}$ its approximation with rates (ε, τ). The symmetric closure $\tilde{R}^s$, which admits (x, y) when $(x, y) \in \tilde{R}$ or $(y, x) \in \tilde{R}$, is a symmetric approximate relation with false-positive rate $1 - (1 - \varepsilon)^2$ and true-positive rate $1 - (1 - \tau)^2$.*

Proof. Because R is symmetric, each unordered pair is tested by two independent trials, on (x, y) and on (y, x) (Axiom 1). The closure admits the pair unless both trials miss. For a negative pair each trial false-positives with probability ε, so the closure false-positives with probability $1-(1-\varepsilon)^2$; for a positive pair each trial is a true positive with probability τ, so the closure accepts with probability $1-(1-\tau)^2$. The closure restores the invariant deterministically, and the same doubling that raises the false-positive rate raises the true-positive rate: at $\tau = 0.9$ the closure true-positive rate is $1 - 0.1^2 = 0.99$. □

The closure is the template for repairing a fragile invariant: replace a conjunction of hopeful trials by an operator that enforces the property outright, and read off the rate it costs.

Single-valuedness

A relation $f \subseteq X \times Y$ is *functional* (single-valued, a partial function) when each key carries at most one value. Approximation breaks this in only one direction. A false negative on a key's true value leaves that key with no value, which is still single-valued; only a *false positive*, which adds a spurious second value, violates functionality. The property survives exactly when no key acquires a spurious value.

Theorem 12.3.3 (Single-valuedness is preserved with vanishing probability)**.** *Let $f \subseteq X \times Y$ be functional. A key with a defined value stays single-valued with probability $(1-\varepsilon)^{|Y|-1}$, since each of the $|Y|-1$ wrong values is an independent false positive at rate ε. Over the d keys with a defined value,*

the probability that $\tilde{f}$ stays single-valued is $(1-\varepsilon)^{d(|Y|-1)}$, which tends to 0 as the value-universe or the domain grows.

Proof. Fix a key x with true value y_x. The $|Y|-1$ pairs (x, y) with $y \neq y_x$ are all negative, each reported with probability ε independently, so x stays single-valued with probability $(1-\varepsilon)^{|Y|-1}$. Losing y_x, an event of probability ω, leaves x with no value and does not violate the property, so it does not enter the product. Distinct keys index disjoint pair sets, so the events are independent and the joint probability is the product over the d defined keys. □

Transitivity

Transitivity is the hardest of the four and, unlike symmetry, has no inexpensive closure. The constraint ranges over *triples*: $(x, y) \in R$ and $(y, z) \in R$ together require $(x, z) \in R$. A single pair participates in many triples, so the implicated trials overlap and the preservation probability does not factor into independent per-triple terms. The honest statement is a limiting one. There are on the order of $|A|^3$ triple constraints, far more than the $|A|$ diagonal constraints of reflexivity or the pair constraints of symmetry, and the trials are coupled, so the preservation probability decays the fastest of the four. Its most fragile part is the set of conclusion pairs $(x, z) \notin R$ that must be *manufactured* by a compensating false positive: as a first-order heuristic, ignoring the cross-triple coupling, each such conclusion contributes a factor near $\varepsilon \ll 1$, and the true probability is smaller still once the coupling is counted. A transitive-closure repair is far more involved than the symmetric one: adding a manufactured pair can create new required conclusions in a cascade with no fixed per-pair cost, so we leave transitivity as the honest limiting result rather than a doubled rate.

The model-order reading

The four results share a reading in the language of Axiom 1. Each invariant partitions the universe into pairs or triples that play different error roles: diagonal versus off-diagonal, positive versus negative, premise versus conclusion. An approximation forced to honor the invariant is therefore a higher-order Bernoulli relation whose blocks carry distinct rates, the same model-order structure projection produced through its varying fan-in (Section 12.2). Element-wise independence keeps the parameter count linear even as the order grows, so the cost of an invariant is charged to the model order, not to an explosion of free rates.

12.4 Approximate Joins in Databases

Section 12.2 treated the join algebraically. This section reads the database *similarity join* through the same framework, showing the published MinHash-LSH analyses to be the chapter-12 join rates specialized to a sketch-based predicate.

The similarity join

For relations R_1, R_2 whose rows are token sets (the words of a document, the shingles of a string), the similarity join at threshold θ is $R_1 \bowtie_\theta R_2 = \{(r_1, r_2) : J(r_1, r_2) \geq \theta\}$, the cross-relation pairs whose Jaccard overlap is at least θ. Record linkage, near-duplicate detection, and entity resolution all reduce to it. The naive evaluation is $|R_1| \cdot |R_2|$ Jaccard computations, prohibitive at scale; the framework reads the join as an approximate relation $\tilde{S}_\theta(r_1, r_2) = [\hat{J}(r_1, r_2) \geq \theta]$ implemented by a sketch.

The MinHash sketch (Definition 10.4.1) is exactly that estimator: the fraction of agreeing signature coordinates is an unbiased estimate of J (Proposition 10.4.1). Build the length-k signature for every row (cost linear in the total token count), and for each candidate pair count agreements and threshold at θ. The two error rates come from the chapter-10 Hoeffding bound (Proposition 10.4.2): a false positive needs $\hat{J}$ to overshoot a sub-θ truth, a false negative needs it to undershoot a super-θ truth, and both are bounded by $\mathrm{P}\left(|\hat{J} - J| \geq t\right) \leq 2\exp(-2kt^2)$ with t the gap to θ. Larger k shrinks both exponentially.

LSH bands

The remaining quadratic factor is candidate enumeration, which LSH (Section 10.4) prunes. Partition the signature into b bands of r rows ($k = br$), hash each band, and call (r_1, r_2) a candidate if any band collides. A band collides with probability J^r, so a pair survives with probability $1-(1-J^r)^b$, the S-curve with inflection $J^* \approx (1/b)^{1/r}$. Placing J^* near θ concentrates candidates on pairs with $J \gtrsim \theta$; the candidate set is sub-quadratic when most pairs have $J \ll \theta$. The pipeline is the intersection of the LSH candidate relation $\tilde{C}$ and the threshold relation $\tilde{S}_\theta$, composing by Theorem 12.2.1; the LSH layer shrinks the false-positive denominator (fewer candidates) and adds to the false-negative side (a truly similar pair can fail to collide), combining by the chapter-3 OR rule.

Remark (The Bloom equijoin)**.** *The equijoin, matching rows on equality of a shared key, admits a different sketch. Build a Bloom filter B_2 over R_2's keys, scan R_1, and for each r_1 whose key hits B_2 scan R_2 for the match. The candidate relation "r_1's key hits B_2" is a one-sided FNR-zero approximate relation (Definition 5.2.1) with per-pair FPR the Bloom rate; an exact verification on each candidate then removes the residual false positives, so the* final *join is exact ($\varepsilon = \omega = 0$) with the Bloom filter serving only as a candidate pruner. This is the framework's certain-reject structure (Chapter 5) deployed as a join accelerator, the equijoin counterpart of the MinHash similarity pipeline.* △

Worked example

Take $|R_1| = |R_2| = 1000$ (so 10^6 candidate pairs), threshold $\theta = 0.5$, and $k = 200$ partitioned as $b = 20$ bands of $r = 10$. The per-pair MinHash threshold is sharp: a pair at $J = 0.3$ is emitted with probability $\leq \exp(-2 \cdot 200 \cdot 0.2^2) = \exp(-16) \approx 1.1 \times 10^{-7}$, and a pair at $J = 0.7$ is missed with the same probability, so away from the boundary the threshold adds negligible error (at $J = \theta$ exactly it is $1/2$).

The LSH layer dominates, and its tuning needs care. With $(b, r) = (20, 10)$ the inflection sits at $J^* = (1/20)^{1/10} \approx 0.74$, *above* the threshold 0.5, making this an aggressive precision-favoring tuning: the survival probability $1 - (1 - J^{10})^{20}$ is ≈ 0.434 at $J = 0.7$, ≈ 0.019 at the threshold $J = 0.5$, and $\approx 1.2 \times 10^{-4}$ at $J = 0.3$. Only 43% of strongly similar pairs and 2% of threshold pairs survive, so the pruning is severe. With a true-similarity distribution of 0.1% at $J \geq 0.5$, 1% at $[0.3, 0.5)$, and 98.9% below, the candidate set is about

$$1000 \cdot 0.434 + 10000 \cdot 0.019 + 989000 \cdot 1.2 \times 10^{-4} \approx 434 + 190 + 119 \approx 743$$

pairs, three orders of magnitude below the 10^6 all-pairs scan: the sub-quadratic win. The cost is the low survival of near-threshold pairs, a real recall penalty; centering J^* on $\theta = 0.5$ (the standard tuning $(1/b)^{1/r} = \theta$, here a larger b) would admit more candidates in exchange for higher recall. The candidate verification is $O(k)$ per pair, $\approx 1.5 \times 10^5$ signature comparisons against the 2×10^8 of the naive scan. This precision/recall trade through (b, r) is the standard operating regime of MinHash-LSH similarity joins. Section 12.5 closes the chapter.

12.5 Bridge

The chapter read one object, the approximate relation, through its operators and its invariants. Section 12.1 cast it as the chapter-5 framework on the pair universe $T \times U$ (Chapter 5), with the chapter-9 binary-function view (Chapter 9) an equivalent reading. Section 12.2 added the relational operators, and they behave sharply differently under approximation: selection is rate-transparent, projection amplifies the false-positive rate by its fan-in and raises the model order while keeping element-wise independence, and a join breaks that independence outright (Theorem 12.2.2), correlating every output tuple built from a shared component. The join's marginal rate (Theorem 12.2.1) is the chapter-6 intersection rate on the triple universe (Chapter 6), the bare product $\varepsilon_1\varepsilon_2$ only in the sparse regime where out-of-join triples are dominated by the both-unrelated region. Section 12.3 then asked which structural invariants survive: reflexivity, symmetry, single-valuedness, and transitivity are each preserved only with a probability that vanishes as the universe grows, the symmetric closure the one cheap repair. Section 12.4 instantiated the machinery on the MinHash-LSH similarity join (Section 10.4), reading its FPR/FNR as the chapter-12 join rates specialized to a sketch-based predicate.

Chapter 13 closes Part III with the type algebra that unifies approximate sets, functions, relations, and sketches into one algebra of Bernoulli[T] constructors. An approximate relation is Bernoulli[bool] on a product universe, equivalently Bernoulli[$T \times U$] in the type-constructor view, and the composition rules each prior chapter derived by direct calculation become functorial properties of the product type. Chapter 12 is one of the first instances of that unification: an approximate relation reread as Bernoulli[$T \times U$] inherits its composition rules from the algebraic structure of the product type rather than from a separate derivation.

Bibliographic Notes

The approximate-relation framework. The relation-level lift of the chapter-5 framework follows Towell [Tow26k], the primary source for the approximate relation as a Bernoulli[bool] on the pair universe and for the join, projection, and selection rate rules of Section 12.2. The chapter reads those rules as the chapter-5 and chapter-6 frameworks specialized to pairs, making Theorem 12.2.1 a corollary of the chapter-6 intersection theorem with cross-relation independence in the role of cross-structure independence.

The classical relational model. The model of a relation as a subset of a Cartesian product, and the operators the chapter approximates, originate with Codd [Cod70], the foundation of the relational-database tradition. The chapter borrows the operators wholesale; the only additions are the approximate lift and its rate analysis.

Probabilistic and uncertain databases. A separate tradition also attaches uncertainty to relational data. Suciu et al. [Suc+11] survey probabilistic databases, which assign each tuple a probability of belonging to the relation and answer queries over the induced distribution of possible worlds; the systems Trio [Wid05] and MayBMS [Hua+09] implement the model. The approximate relation here is a different object. Its uncertainty is not a probability that a stored fact is true but a per-query error rate of a fixed approximate structure: the relation is exact and known, and the randomness lives in the membership test, with the rates ε and ω describing how a Bernoulli trial reports a definite truth. Possible-worlds semantics ask which tuples are probably present; the approximate-relation semantics ask, for a given tuple, how reliably its membership comes back. The two are complementary rather than competing, one modeling uncertain data tested exactly, the other certain data tested approximately.

Similarity-join algorithms. The algorithmic instance of Section 12.4 combines Broder [Bro97]'s MinHash with Indyk and Motwani [IM98]'s locality-sensitive hashing, systematized as a database operator for set-similarity joins by Sarawagi and Kirpal [SK04]; the many later variants (prefix, positional, and length filtering) are each an alternative implementation of the same approximate relation $\tilde{S}_\theta$, with rates by the same chapter-12 machinery. Theorem 12.2.1 commits to no particular sketch: the chapter-10 MinHash bound is used only because the book already developed it, and the Bloom equijoin of Section 12.4 is another instance with different per-pair rates.

Chapter 13

The Algebra of Approximate Types

13.1 Algebraic Types Recap

Twelve chapters in, the same calculations keep recurring. The Kronecker product of channel matrices that organized m Boolean queries in chapter 4 returned for the n-ary channels of chapter 9 and the relational products of chapter 12; the set-composition rules of chapter 6 lined up with chapter 9's; CountMin and HyperLogLog were noisy counters, a different output type under the same two axioms. The thread is that each structure is a Bernoulli$[T]$ for some algebraic type T. This chapter is the unification.

The four constructors. Algebraic data types are built from four constructors: the *product* $T \times U$ (pairs, records, structs; cardinality $|T||U|$), the *sum* $T + U$ (tagged unions, variants, enums; cardinality $|T| + |U|$, with a tag recording which summand), the *list* LT (finite sequences of any length), and the *function* $T \to U$ (the most general, into which the others embed). They close under composition, so any type built from primitives and these four has a clear structural decomposition.

Each approximate structure so far lives at one constructor. An *approximate set* (Chapter 5) is a noisy predicate Bernoulli$[T \to \text{bool}]$, with the chapter-6 composition rules (Chapter 6) the rules for composing such predicates. An *approximate map or channel* (Chapter 9) is a Bernoulli$[T \to U]$ for finite T, U, the n-ary channel. A *sketch* (Chapter 10) is a noisy value of an aggregate type, CountMin and HyperLogLog as Bernoulli$[\mathbb{N}]$, MinHash as Bernoulli$[[0, 1]]$. An *approximate relation* (Chapter 12) is a noisy

predicate on a product, Bernoulli$[T \times U \to \text{bool}]$. The type tells you which composition rules apply.

The lift. Write Bernoulli$[-]$ as a type-level operation: Bernoulli$[T]$ is the type of random approximate values over T, with the atom Bernoulli[bool] the noisy bit of Part I. The substantive content of the chapter is how Bernoulli$[-]$ behaves on each constructor. Section 13.2 shows it factors cleanly through three of the four, the product (a tensor of factor lifts), the list (a per-position lift), and the function (a per-input lift), each by the Kronecker factorization of chapters 4 and 9. The sum is the exception: a sum value's *tag* can itself err, coupling the two components, and there the factorization breaks. Section 13.3 packages the three clean rules as a single structure-preserving map and locates the sum's tag error as the place where the framework's atomic false-positive and false-negative rates come from.

13.2 Bernoulli$[T]$ for Composite T

This section gives the lift Bernoulli$[-]$ on each constructor. Three of the four factor cleanly; the sum is the exception.

Product types: the Kronecker isomorphism

A noisy pair is a pair of noisy values. If $\tilde{t}$ has channel Q_T and $\tilde{u}$ has channel Q_U, with independent noise, the joint channel at each (t, u) is the product of the marginals, so the joint matrix is the Kronecker product $Q_T \otimes Q_U$, the chapter-4 and chapter-9 factorization (Theorem 4.4.1, Theorem 9.3.1) read at the type level.

Theorem 13.2.1 (Product-type isomorphism)**.** *For types T, U with independent noise channels,* Bernoulli$[T \times U] \cong$ Bernoulli$[T] \otimes$ Bernoulli$[U]$*, the Kronecker product of the component channels: a $K_T K_U \times L_T L_U$ matrix with $(Q_T \otimes Q_U)_{(j_T, j_U),(\ell_T, \ell_U)} = (Q_T)_{j_T, \ell_T} (Q_U)_{j_U, \ell_U}$.*

Axiom 1 factors the joint into a product across components and Axiom 2 keeps each row stable, so the proof is the chapter-9 calculation with the two factors now of different dimensions. The parameter count is additive: without independence the joint would need $K_T K_U (L_T L_U - 1)$ parameters, but under it only $K_T(L_T - 1) + K_U(L_U - 1)$, the parameters of Q_T plus those of Q_U. Products are the well-behaved case, where approximation distributes and parameters grow additively.

Sum types: the tag error breaks the factorization

A sum $T + U$ tags each value Left(t) or Right(u). An approximate sum value errs in two ways: a *tag error* flips the tag with probability δ, and a *payload error* corrupts the value within the correct component at rate ε_T or ε_U. For a latent Left(t),

$$\mathrm{P}\left(\widetilde{\mathrm{Left}(t)}\right) = (1-\delta)(1-\varepsilon_T), \quad \mathrm{P}\left(\mathrm{Left}(t' \neq t)\right) = (1-\delta)\tfrac{\varepsilon_T}{|T|-1}, \quad \mathrm{P}\left(\mathrm{Right}(\cdot)\right) = \delta,$$

the last spread over the wrong component.

Proposition 13.2.1 (Sum types do not factor)**.** Bernoulli[$T+U$] $\ncong$ Bernoulli[T]$\oplus$ Bernoulli[U] *in general: the confusion matrix of an approximate sum does not factor as a direct sum of per-component channels.*

The reason is structural and is the heart of the product-versus-sum distinction. In a product, both components are always active and independent, so the channel tensors. In a sum, the tag error *couples* the components: when the tag is correct only one component's payload channel applies, and when it is wrong the value is drawn from the *other* component entirely. The block-diagonal form Bernoulli[T] $\oplus$ Bernoulli[U] (one payload channel per summand, zero off-diagonal) is only the $\delta = 0$ degenerate case, a sum with an exact tag; any tag noise puts mass in the off-diagonal blocks and no factorization recovers it. This is Core Principle 2: products preserve parametric parsimony, sums break it.

The break is not a peripheral failure. The optional type Maybe(T) = T+ Unit adds a "nothing" case, and its tag error *is* approximate set membership: a tag flip on Just(t) produces Nothing, a false negative, and a flip on Nothing produces some Just(t), a false positive, with $\delta = \varepsilon$ for negatives and $\delta = \omega$ for positives. The sum's tag error is exactly the atomic membership error the whole framework rests on, which is why the sum constructor cannot be made to factor: factoring it away would factor away the framework's irreducible rate.

List and function types

A list of known length n is the product T^n, so Bernoulli[T^n] $\cong$ Bernoulli[T]$^{\otimes n}$, the Kronecker power of the per-position channel, with parameter count $K_T(L_T - 1)$ independent of n. A variable-length list, however, is a *sum* over lengths, $LT \cong \sum_{n\geq 0} T^n$, so a noisy length is a sum-type tag error over the length index: Bernoulli[LT] is a mixture over the noisy length (a

Bernoulli[$\mathbb{N}$]), each component a per-position Kronecker product, and it does not reduce to a single clean product. A HyperLogLog cardinality estimate is exactly this noisy length, composing with per-element noise by the mixture rule, not by a simple per-element product.

A noisily approximated function $f : T \to U$ produces one Bernoulli[U] per input, so Bernoulli[$T \to U$] $\cong T \to$ Bernoulli[U] $\cong \prod_{t \in T}$ Bernoulli[U], the chapter-9 framework: the channel is a function from input blocks to output distributions. This factors over inputs as a Kronecker product of $|T|$ per-input channels, with parameter count $|T|(L_U - 1)$, growing with the domain. Approximate sets ($U =$ bool) and relations (T a product) are sub-cases.

Constructor	Lift	Channel form
$T \times U$	Bernoulli[T] $\otimes$ Bernoulli[U]	Kronecker product
LT	Bernoulli[T]$^{\otimes n}$ (mixture if length noisy)	Kronecker power
$T \to U$	$T \to$ Bernoulli[U]	per-input Kronecker
$T + U$	does *not* factor (tag coupling)	not block-diagonal unless $\delta = 0$

The first three rows are forced by Axioms 1 and 2 at the type-structural level; the fourth records where those axioms' independence fails. Section 13.3 packages the three clean rules.

Worked example

A 3-class classifier reporting a class label and a times-seen count has output type $\{\text{cat}, \text{dog}, \text{bird}\} \times \mathbb{N}$, a product of a 3-element enumeration (governed by a 3×3 confusion matrix Q_{class}) and a count (a CountMin-style Bernoulli[$\mathbb{N}$], Q_{count}). Under independence the joint factors as Bernoulli[$\{\text{cat}, \text{dog}, \text{bird}\}$]$\otimes$ Bernoulli[$\mathbb{N}$], the Kronecker product $Q_{\text{class}} \otimes Q_{\text{count}}$, with additive parameter count $3 \cdot 2 + K_{\mathbb{N}}(L_{\mathbb{N}} - 1)$. To compose two such classifiers one applies the chapter-12 join rate to the label and the chapter-10 sketch composition to the count, with no cross-interaction: the type structure says where each rule goes. Composing approximate structures, which used to need a per-structure derivation, reduces to applying the product, list, and function rules to the components, with the sum flagged as the place a tag error couples them.

13.3 The Composition Functor

Section 13.2 gave the lift on each constructor. This section observes that the rules belong together as one structure-preserving map, with the sum as

the documented exception.

A structure-preserving map

The lift Bernoulli[−] takes a type-level operation to a channel-level one. The product becomes a Kronecker product (Theorem 13.2.1), the list a Kronecker power, the function a per-input Kronecker product. A map that takes operations to corresponding operations is a *structure-preserving map*; the formal categorical name is *functor* (§13.5). The four cases:

Type level	Channel level	Reading
$T \times U$	Bernoulli[T] $\otimes$ Bernoulli[U]	Kronecker product
LT	LBernoulli[T] (mixture if length noisy)	per-position lift
$T \to U$	$T \to$ Bernoulli[U]	per-input lift
$T + U$	does *not* factor	tag coupling (Proposition 13.2.1)

The first three rows are isomorphisms: the type and the channel specify the same Bernoulli[−] instance up to relabeling, and Bernoulli[−] commutes with the product, list, and function constructors. The fourth row is the exception, the sum, where the tag error couples the components and no factorization recovers the channel. That exception is not a gap in the theory but its foundation: the optional type's tag error is the atomic membership rate (§13.2).

Remark (Bernoulli[−] as a monad)**.** *The function row* Bernoulli[$T \to U$] $\cong$ $T \to$ Bernoulli[U] *is the signature of a* monad*. The Kleisli arrow, a noise-introducing function* $T \to$ Bernoulli[U]*, takes a pure value to a noisy one; two arrows* $f : T \to$ Bernoulli[U] *and* $g : U \to$ Bernoulli[V] *compose into* $T \to$ Bernoulli[V] *by feeding* f*'s output through* g *and averaging the intermediate noise (the monad's bind); and a pure value embeds as the point distribution concentrated on it (the unit). These three make* Bernoulli[−] *a monad in the programming-language sense, the noisy-value counterpart of the optional and list monads, with the formal axioms in Wadler [Wad92].* △

Why it matters

Without the table, every approximate structure needed its own composition derivation: chapter 6 for sets, chapter 10 for sketches, chapter 12 for relations. With it, composition is mechanical for any type built from the three factoring constructors: decompose the type and apply the matching

row at each level. Adding a new structure becomes a matter of identifying its type, not redoing the algebra, a sketch estimating matrix entries is Bernoulli$[T \times U \to \mathbb{R}]$, and the function and product rows determine its composition with no fresh derivation.

The structures already built each instantiate the table. An approximate set (Chapter 6) is Bernoulli$[T \to \text{bool}]$, so the function row gives the composition rule and chapter-6's results are the function row at codomain bool, read pointwise. A CountMin sketch (Chapter 10) is Bernoulli$[T \to \mathbb{N}]$, its additivity the product-row Kronecker structure on the codomain. An approximate relation (Chapter 12) is Bernoulli$[T \times U \to \text{bool}]$, combining the function row (per-input lift) and the product row (on the domain). The n-ary channel of Chapter 9 is the building block $X \to Y$ underneath them all, distinguished only by codomain and domain structure.

The framework as one thing

There is one underlying object across the book. The atom is the noisy bit Bernoulli[bool]; every other structure is a Bernoulli$[T]$ for an algebraic type T built from bool and the constructors, and the composition rules that organized the earlier chapters are the rows of the table. The title's "algebra of approximate computation" is the algebra of the type constructors, with Bernoulli$[-]$ its action on the approximate side, factoring through the product, list, and function and meeting its irreducible limit at the sum's tag error. The remaining chapters specialize the algebra rather than extend it: chapter 14 builds approximate sets at the entropy floor, chapter 15 the perfect-hash filters, chapter 16 catalogues where the independence assumptions break, and chapters 17 and 18 turn the algebra to cryptographic use. Each is a specialized application of the table.

13.4 Closing Part III

Three parts are now in place. Part I built the atom, the noisy bit Bernoulli[bool] with its 2×2 channel and rates (ε, ω). Part II specialized it one direction, to predicates on a universe, and worked out the algebra of approximate sets, their composition (chapter 6), classification measures (chapter 7), and information-theoretic floor (chapter 8). Part III specialized it a second direction, to maps and types: Chapter 9 lifted the binary channel to $K \times L$ channels, Chapter 10 to streaming counters and estimators, chapter 11 to randomized deciders, Chapter 12 to predicates on pairs, and this chapter unified the four into one type-driven algebra.

Each part widens the type structure. Part I fixes the codomain to bool and studies the atom; Part II keeps the codomain bool and studies functions $T \to$ bool; Part III releases both domain and codomain to range over the algebraic types built from bool and the four constructors. The two rates of Part I become the channel matrix of Part III, and the union and intersection of Part II become rows of the table in Section 13.3. Products go to Kronecker products, lists to per-position lifts, functions to per-input lifts; the sum is the exception, where the tag error couples the components and the factorization breaks, the same tag error that is the atomic membership rate the whole framework rests on.

Three parts remain. Part IV gives the constructive payoff Part II's lower bound left open: Chapter 14 builds the Bernoulli Hash Function, an approximate set meeting the chapter-8 entropy floor by abandoning the shared bit array for a perfect-hash fingerprint encoding, and Chapter 15 builds the FPR-zero perfect hash filters on the other one-sided axis. Part V draws the boundary and turns the algebra to cryptographic use: Chapter 16 catalogues where the independent-error model fails (correlated errors, drifting rates, adversarial queries), and chapters 17 and 18 read the algebra in reverse, the salt of a Bernoulli hash function as a key and the acceptance predicate as a cipher map, with the chapter-8 floor governing both space cost and security budget. Chapter 19 then closes the book by re-reading the whole framework in operator form, elements as basis vectors and noise as a stochastic matrix, where the Kronecker products of Section 13.3 become the formal tensor factorization of an independent set's noise. With Part III closed, the algebra is in place; the next parts build on it.

Bibliographic Notes

The type-driven algebra. The composition table of Section 13.3 and the type-by-type lift of Section 13.2 follow Towell [Tow26j], "Random approximate values over algebraic types," which works out the algebraic data type of Bernoulli[T] and the constructor rules, including the sum-type tag error and its non-factorization. That manuscript treats Bernoulli[$-$] as a type-level operation with the channel-matrix derivations of chapters 4 and 9 as consequences; the book takes the opposite route, starting from the channel matrix and reading the type algebra off it. The two agree.

Algebraic data types. The four-constructor view (products, sums, lists, functions) is the standard algebraic-data-type framing of programming-language

theory, the type system of ML and Haskell descended from Hindley–Milner. Hudak [Hud89]'s survey is a readable history of how it grew out of the typed lambda calculus, for readers coming from imperative backgrounds.

The categorical reading. The structure-preserving language of Section 13.3 is informal category theory: Bernoulli[−] is a *functor* from algebraic types to noisy channels, and it is monoidal with respect to the product and exponential constructors (which it sends to Kronecker product and per-input Kronecker factorization) but *not* the sum, where the tag error couples the components and the direct-sum structure breaks. The function-row identity Bernoulli$[T \to U] \cong T \to$ Bernoulli$[U]$ makes Bernoulli[−] a monad (Section 13.3). Mac Lane [Mac98] is the standard reference for the machinery (functors, natural transformations, monads), and Wadler [Wad92] translates the categorical monad into the programming-language idiom used here. Probability monads in this sense go back to Giry and Lawvere and have re-emerged as probabilistic-programming DSLs; the Bernoulli lift is the one whose channel representation is a finite stochastic matrix with closed-form composition. The book stops at the table because that is the level the practitioner needs.

Part IV

Optimality and Construction

Chapter 14

Optimal Bernoulli Sets (BHF)

14.1 The Optimality Question

Chapter 8 closed Part II with a contrast it could not resolve. The entropy floor (Theorem 8.3.1) certified that some FNR-zero representation could in principle reach $-\log_2 \varepsilon$ bits per element, but it exhibited none; the standard Bloom filter, the only concrete FNR-zero scheme on the table, sat a fixed multiplicative factor above that floor. The bound was diagnostic, not constructive. This chapter supplies the construction: an explicit FNR-zero approximate set that meets the floor up to integer rounding and a sublinear correction, closing the gap the Bloom filter leaves open.

The gap

Theorem 8.3.1 of Section 8.3 states that any FNR-zero approximate-set representation with false-positive rate at most ε requires

$$L/n \;\geq\; -\log_2 \varepsilon \;+\; o(1)$$

bits per element in the asymptotic regime $N \gg n$, ε fixed. The argument is a counting bound: 2^L representations must cover $\binom{N}{n}$ possible target sets, and each representation describes at most $\binom{|A_R|}{n}$ true sets inside its accepted set, forcing L to grow linearly in $n \log_2(1/\varepsilon)$.

Proposition 8.4.1 of Section 8.4 computes the Bloom filter's budget from Definition 5.2.1: at the optimal hash count $k^* = (m/n)\ln 2$, the per-element cost is

$$m/n \;=\; \log_2(1/\varepsilon) \cdot \log_2 e,$$

or $\log_2 e \approx 1.4427$ times the floor, a constant independent of ε. The overhead traces to a structural choice: k independent hashes write into a flat shared bit array, and the analysis maximizes per-bit entropy by tuning the array to half occupancy. Half-occupancy is the wrong target. Bits should pay for the discrimination the structure performs, not for the encoding's internal balance.

The constructive question

Section 8.4 isolated three Bloom design choices, k independent hashes, a flat shared bit array, and a conjunction query, and observed that closing the gap requires abandoning at least one. Which one, and what replaces it, the remark left open. A construction that answers it needs four properties: FNR-zero by construction, so it stays on the perfect-filter axis with the Bloom filter; an FPR provably at most ε; a per-element budget matching $-\log_2 \varepsilon$ up to a sublinear correction; and a query running in time comparable to Bloom's. The first two are correctness, the third is the optimality the chapter promises, the fourth is what makes the construction deployable rather than an existence proof. The floor theorem guarantees that some encoding meets all four asymptotically; the question is whether an explicit one does, and what it looks like.

The answer in sketch

The Bernoulli Hash Function (BHF) abandons the flat shared bit array. In its place it uses a *perfect hash* $\phi : A \to [n]$ that bijects the designated set A onto $[n] = \{1, \ldots, n\}$, and a *fingerprint table* $F : [n] \to \{0,1\}^r$ storing $F[\phi(a)] = g(a)$ for each $a \in A$, where $g : U \to \{0,1\}^r$ is a secondary hash and $r = \lceil -\log_2 \varepsilon \rceil$ is the fingerprint width. A query for x computes the slot $\phi(x)$, reads the stored fingerprint $F[\phi(x)]$, and returns $\top$ exactly when it equals $g(x)$.

For $x \in A$ the perfect hash sends x to its own slot, where the build pass stored $g(x)$; the comparison always succeeds, so no false negatives occur. For $x \notin A$ the slot holds $g(a)$ for whichever $a \in A$ shares the slot, and $g(a) = g(x)$ with probability 2^{-r} under the uniformity of g; with $r = \lceil -\log_2 \varepsilon \rceil$ the false-positive rate is at most ε by construction. The space cost is nr bits for the table plus a sublinear perfect-hash term, so the per-element budget is

$$r \;+\; o(1) \;=\; -\log_2 \varepsilon \;+\; O(1),$$

matching the floor up to the integer rounding of r. The construction trades Bloom's flat array for a perfect-hash-addressed table, and the per-element

cost drops from $\log_2 e \cdot \log_2(1/\varepsilon)$ to $\log_2(1/\varepsilon)$ plus rounding. Section 14.2 gives the construction in full, Section 14.3 the bit budget and the optimality theorem, and Section 14.4 the false-positive bound.

14.2 The BHF Construction

Section 14.1 sketched the answer in two ingredients and two algorithms. This section states the construction in full: a definition, pseudocode, the structural argument for $\omega = 0$, a preview of the false-positive analysis that Section 14.4 completes, and a worked example placing the BHF's bit budget beside the Bloom filter's at the same target.

The two ingredients

Fix a finite universe U and a designated set $A \subseteq U$ with $|A| = n$. The construction depends on two hash functions of opposite character.

(i) The perfect hash ϕ. A *perfect hash*

$$\phi \;:\; A \longrightarrow [n], \qquad [n] = \{1, 2, \ldots, n\},$$

is injective on A, hence a bijection between A and the addressing range. It is not a hash function in the usual sense of mapping a large universe uniformly into a small range; it is data-dependent, constructed at build time from A so that no two elements of A collide. Its construction cost and storage are computed in Section 14.3 and Section 14.5; for now ϕ is a black-box bijection $A \to [n]$, extended by some rule to the whole universe (the extension does not affect the FPR bound).

The departure from the Bloom design is exactly this collision-freedom. An ordinary hash collides at the rate the range-to-universe ratio allows, and the Bloom filter accommodates collisions by spreading each insertion across k independent hashes, the k-way conjunction at query time absorbing the per-hash noise. The BHF instead commits to A at build time and constructs a hash bespoke to A, collision-free by design. With no per-hash noise to absorb, a single fingerprint comparison replaces the k-way conjunction. Building ϕ is the workload the BHF moves from query time, where Bloom pays for it in extra hash evaluations and bits, to build time, where it is paid once and amortized.

(ii) The fingerprint table F and hash g. The *fingerprint table*

$$F \;:\; [n] \longrightarrow \{0,1\}^r$$

holds n slots of r bits each, with fingerprint width

$$r \;=\; \lceil -\log_2 \varepsilon \rceil, \tag{14.1}$$

the smallest integer for which $2^{-r} \le \varepsilon$. It is populated by a *fingerprint hash* $g : U \to \{0,1\}^r$ assumed uniform on its codomain, instantiated in practice as the low r bits of SHA-256(x), a truncated MurmurHash3, or any family known to behave uniformly on the application's inputs. Uniformity of g is the only hash-function assumption the FPR analysis of Section 14.4 needs.

The two hashes play complementary roles. ϕ never collides on A and addresses the table; g collides at the controlled rate 2^{-r} and supplies the fingerprint that discriminates members from non-members. The width r is the only place ε enters the construction: tightening the target from ε to ε^2 adds one bit to r and doubles the table, the relation the entropy floor predicts.

Construction and query

Given A, ϕ, and g, the table is built in one pass over A:

$$\text{for each } a \in A : \quad F[\phi(a)] \;\coloneqq\; g(a). \tag{14.2}$$

Bijectivity of ϕ on A writes each slot exactly once, so the table is fully populated and records $g(a)$ at the slot $\phi(a)$ reserved for each a.

Definition 14.2.1 (Bernoulli Hash Function)**.** *The* Bernoulli Hash Function *(BHF) for a designated set A with parameters ϕ, g, and $r = \lceil -\log_2 \varepsilon \rceil$ is the pair (ϕ, F) together with the membership query*

$$\textsc{Query}(x) \;=\; \top \;\; \textit{iff} \;\; F[\phi(x)] = g(x).$$

After the construction pass (14.2)*, $x \mapsto \textsc{Query}(x)$ is the membership report of the approximate set $\mathsf{A}^{\pm}$.*

The data structure stores the perfect hash and the table; the query reads one slot and compares two r-bit strings. There is no flat bit array, no k-way conjunction, no occupancy parameter. Where the Bloom filter spreads information about A thinly across a shared array, the BHF concentrates it: each element owns one slot holding the fingerprint that distinguishes it from non-members. That concentration is what closes the gap. In the Bloom filter, per-bit entropy is the quantity to optimize, and its maximum at half occupancy costs $\log_2 e$ times the floor; in the BHF the bit budget is

Algorithm 6: BHF construction and query

```
1  procedure BUILD(A)
2  |   construct perfect hash φ : A → [n];
3  |   allocate fingerprint table F of size n, each slot r bits;
4  |   for each a ∈ A do
5  |   |   F[φ(a)] ← g(a);
6  |   end
7  |   return (φ, F);
8  function QUERY(x)
9  |   i ← φ(x);
10 |   if F[i] = g(x) then return ⊤;
11 |   return ⊥;
```

set directly by the width r, which equals the floor's $-\log_2 \varepsilon$ up to integer rounding, and the per-bit optimization simply does not arise.

Algorithm 6 writes the construction and query in pseudocode.

BUILD runs in one pass over A plus the cost of constructing ϕ; QUERY runs in time dominated by two hash evaluations and an r-bit comparison, constant in n. Against the Bloom pseudocode of Algorithm 2, whose QUERY evaluates k hashes and reads k bits, the BHF replaces k hash evaluations with 2 and the k-way conjunction with a single equality test, a query-time factor of $k/2 \approx (m/n) \ln 2/2 \approx 0.347(m/n)$ at the Bloom optimum. The price is that the construction is static (no in-place insertions) and the perfect hash takes $O(n)$ to build; Section 14.5 treats the trade-off.

The false-negative rate is exactly zero

The argument mirrors the Bloom case and is shorter. Suppose $x \in A$. The build pass (14.2) set $F[\phi(x)] = g(x)$, and the table is never modified afterward (the BHF is static), so $F[\phi(x)] = g(x)$ at every query time. QUERY(x) reads slot $\phi(x)$, finds $g(x)$, recomputes $g(x)$, and returns $\top$. The argument rests on three facts: the build pass wrote $g(\cdot)$ into slot $\phi(\cdot)$, the table is immutable, and ϕ is a function, so the query-time slot $\phi(x)$ equals the build-time slot. Hence

$$\omega \;=\; \mathrm{P}\left(\textsc{Query}(x) = \bot \mid x \in A\right) \;=\; 0$$

for every $x \in A$ and every parameter setting. The BHF is one-sided, on the same axis of the perfect-filter line as the Bloom filter; its only failure mode

is the false positive.

False-positive rate, preview

For $x \notin A$, the slot $\phi(x)$ equals $\phi(a^*)$ for the unique $a^* \in A$ mapped there, so the query reads $F[\phi(x)] = g(a^*)$ and compares it to $g(x)$. By uniformity of g, the strings $g(a^*)$ and $g(x)$ are independent uniform on $\{0,1\}^r$, agreeing with probability

$$\varepsilon_{\mathrm{BHF}} \;\le\; 2^{-r} \;=\; 2^{-\lceil -\log_2 \varepsilon \rceil} \;\le\; \varepsilon.$$

Section 14.4 develops the argument, including the role of the perfect-hash extension to $U \setminus A$; the bound above suffices for the comparison below.

A worked example

Fix $n = 1000$ and target $\varepsilon = 0.01$. The fingerprint width is

$$r \;=\; \lceil -\log_2 0.01 \rceil \;=\; \lceil 6.644 \rceil \;=\; 7 \text{ bits},$$

since $2^{-7} \approx 0.0078 \le 0.01 \le 2^{-6} \approx 0.0156$. The table occupies $nr = 7000$ bits (875 bytes); adding a modern perfect-hash representation, which contributes a few bits per element at $n = 1000$ (Section 14.3), brings the BHF total to roughly 9000 bits. The Bloom filter at the same target is sized by Proposition 8.4.1 at

$$m \;=\; n \cdot \log_2(1/\varepsilon) \cdot \log_2 e \;=\; 1000 \cdot 6.644 \cdot 1.4427 \;\approx\; 9586 \text{ bits}.$$

At $n = 1000$ the two occupy similar space, the perfect-hash correction nearly cancelling the fingerprint saving. The picture changes at scale: the perfect-hash term per element shrinks below one bit, leaving the BHF at $r = 7$ bits against the Bloom filter's 9.59, the full 2.6-bit advantage the floor allows. Against the floor itself, $-\log_2 0.01 = 6.644$, the BHF's 7 bits exceed it by the integer-rounding gap of 0.356 bits, while the Bloom filter sits ≈ 2.95 bits above. Section 14.3 works the comparison at $n = 10^6$ and tabulates it across target rates.

14.3 Space Analysis

The BHF stores two objects, the perfect hash ϕ and the fingerprint table F, and the total bit budget is their sum. The fingerprint table is the dominant term and matches the floor of Theorem 8.3.1 up to the integer rounding of r; the perfect-hash representation is a sublinear correction that vanishes

asymptotically. Together they give the chapter's headline: the BHF reaches the floor up to one bit of rounding plus an $o(1)$ term, and the Bloom filter's $\log_2 e$ multiplicative overhead is gone.

Fingerprint storage

The table $F : [n] \to \{0,1\}^r$ holds n slots of r bits, so

$$L_F \;=\; n \cdot r \;=\; n \cdot \lceil -\log_2 \varepsilon \rceil \tag{14.3}$$

bits, with r from (14.1). The dependence on the target rate is logarithmic, each base-10 digit of precision costing about 3.32 bits per element, and the dependence on set size is linear. There is no padding and no per-slot overhead beyond the fingerprint: the table is a flat array addressed by ϕ, the densest representation of one uniform fingerprint per element. The Bloom filter, by contrast, spreads each insertion across k bits and pays the half-occupancy load factor that the optimal-k analysis selects; the BHF avoids both penalties by resolving collisions at build time.

The integer rounding in r contributes between 0 and 1 bits per element over the real-valued floor $-\log_2 \varepsilon$, since $\lceil x \rceil - x \in [0, 1)$, averaging 0.5 bits over a logarithmically uniform ε. This additive, bounded overhead is the structural contrast with the Bloom filter's $\log_2(1/\varepsilon) \cdot \log_2 e$ budget (Proposition 8.4.1), whose overhead is multiplicative and grows with $\log_2(1/\varepsilon)$. As $\varepsilon \to 0$ the BHF budget approaches the floor; the Bloom budget stays 44% above it.

Perfect-hash storage

The perfect hash ϕ is itself a stored structure, since queries need it to address slots. Its size depends on the construction. The BDZ family of Belazzougui, Botelho, and Dietzfelbinger (§14.7) achieves

$$L_\phi \;=\; O\big(n \log\log\log n / \log n\big) \tag{14.4}$$

bits, which is $o(n)$: the per-element overhead $L_\phi/n = O(\log\log\log n / \log n)$ tends to zero. Earlier constructions (Fredman-Komlos-Szemeredi, Hagerup-Tholey, the displacement-based families) reach at most $O(n)$ bits with leading constants of 2 to 4 bits per element. At realistic scales (n between 10^3 and 10^9) a BDZ-style construction occupies on the order of 2 to 3 bits per element, and the present accounting takes (14.4) as the bound with the constant treated as a small additive correction.

At $n = 10^6$ the asymptotic form evaluates to roughly 0.2 to 0.5 bits per element in practical implementations, against a fingerprint term of r bits ranging from 7 at $\varepsilon = 10^{-2}$ to 20 at $\varepsilon = 10^{-6}$; the perfect-hash overhead is a few percent of the fingerprint cost and shrinks at larger n, below 0.1 bits per element by $n = 10^9$. At deployment scale it is a rounding correction to the fingerprint term, not a separate concern.

Total bits per element

Combining the two terms,

$$L_{\mathrm{BHF}} \;=\; L_F + L_\phi \;=\; n \cdot \lceil -\log_2 \varepsilon \rceil \;+\; O\big(n \log\log\log n / \log n\big), \tag{14.5}$$

and dividing by n,

$$L_{\mathrm{BHF}}/n \;=\; \lceil -\log_2 \varepsilon \rceil \;+\; O(\log\log\log n / \log n). \tag{14.6}$$

The first term lies between $-\log_2 \varepsilon$ and $-\log_2 \varepsilon + 1$; the second is $o(1)$.

Theorem 14.3.1 (BHF optimality)**.** *The Bernoulli Hash Function of Definition 14.2.1 on a designated set A with $|A| = n$ and target $\varepsilon \in (0, 1)$ uses*

$$L_{BHF}/n \;=\; \lceil -\log_2 \varepsilon \rceil \;+\; o(1) \;=\; -\log_2 \varepsilon \;+\; O(1)$$

bits per element, achieving the entropy floor of Theorem 8.3.1 up to one bit of integer rounding plus a sublinear correction. The multiplicative overhead $\log_2 e \approx 1.4427$ of the standard Bloom filter (Proposition 8.4.1) is absent: the BHF's overhead is additive and bounded by a constant.

The theorem is the constructive complement to Theorem 8.3.1. The floor said no FNR-zero representation uses fewer than $-\log_2 \varepsilon$ bits per element asymptotically; the present theorem exhibits one using no more than that plus an additive constant. Together they pin the per-element cost of a sufficient FNR-zero construction to $[-\log_2 \varepsilon + o(1),\, -\log_2 \varepsilon + 1 + o(1)]$, with the BHF at the top of the interval and the slack the unavoidable cost of a whole number of fingerprint bits.

The optimality can be sharpened from "matches the floor" to "optimal within its structural class."

Remark (Optimality among perfect-hash-addressed structures)**.** *Consider any FNR-zero structure of the form (ϕ, F) with $\phi : A \to [n]$ a perfect hash and $F : [n] \to \{0,1\}^r$ a fingerprint table. For a non-member x mapped to slot i, the verdict is $\top$ iff $F[i]$ matches the r-bit string the query derives from*

x; under any non-adversarial query distribution with a uniform fingerprint derivation, the match probability is exactly 2^{-r}*, so* $\varepsilon \geq 2^{-r}$*. With the floor's* $r \geq -\log_2 \varepsilon$*, the achievable* (r, ε) *region is* $\varepsilon \geq 2^{-r}$ *and* $r \geq \lceil -\log_2 \varepsilon \rceil$*. The BHF sits at the corner,* $r = \lceil -\log_2 \varepsilon \rceil$ *and* $\varepsilon = 2^{-r} \leq \varepsilon$*: no perfect-hash-addressed FNR-zero structure meets the same target with fewer bits per element.* △

Comparison table

Fix $n = 10^6$. Table 14.1 compares the entropy floor $-\log_2 \varepsilon$, the Bloom filter at $\log_2(1/\varepsilon) \cdot \log_2 e$ bits per element, and the BHF at $\lceil -\log_2 \varepsilon \rceil$ (ignoring the sublinear perfect-hash term, a fraction of a bit at $n = 10^6$).

Table 14.1: Bits per element at $n = 10^6$. Floor is the bound of Theorem 8.3.1; Bloom is the standard construction (Proposition 8.4.1); BHF is the optimum of Theorem 14.3.1, excluding sublinear perfect-hash overhead.

ε	Floor	Bloom	BHF
10^{-2}	6.64	9.59	7
10^{-3}	9.97	14.38	10
10^{-6}	19.93	28.76	20

The Bloom filter pays a multiplicative $\log_2 e \approx 1.44$ at every row; the BHF pays an additive $\lceil -\log_2 \varepsilon \rceil - (-\log_2 \varepsilon)$, concretely $0.36, 0.03, 0.07$ bits. The Bloom-versus-BHF gap grows with $-\log_2 \varepsilon$, from 2.59 bits per element at $\varepsilon = 10^{-2}$ to 8.76 at $\varepsilon = 10^{-6}$; the BHF-versus-floor gap stays below one bit. At $n = 10^6$, $\varepsilon = 10^{-6}$, the absolute saving over Bloom is $8.76 \cdot 10^6$ bits, about 1.1 megabytes.

The $\lceil \cdot \rceil$ rounding

The budget is $\lceil -\log_2 \varepsilon \rceil$ rather than $-\log_2 \varepsilon$ because r is an integer; there is no fractional bit in a table addressed by the perfect hash. The rounding is not wasted silently: at integer r the achieved FPR is 2^{-r}, at most the target but generally strictly below it. Writing $-\log_2 \varepsilon = k + \delta$ with k integer and $\delta \in [0, 1)$, the width is $r = k + 1$ when $\delta > 0$, and the achieved FPR is $2^{-r} = \varepsilon / 2^{1-\delta}$, smaller than ε by a factor between 1 (as $\delta \to 1$) and 2 (as $\delta \to 0^+$). The BHF over-achieves the target by the rounding slack, on average about half.

The slack can in principle be recovered: arithmetic or block coding across k elements drives the per-element cost toward $-\log_2 \varepsilon$ without in-

teger rounding, at the cost of query time (a fractional-bit fingerprint cannot be compared at machine-word speed) and implementation complexity. In practice the ≤ 1 bit of waste is accepted and the BHF is deployed with integer r. The construction is asymptotically optimal up to that rounding, which is small and well-understood, and on those terms the BHF is the constructive answer to the chapter-8 floor question.

14.4 False-Positive Rate Analysis

Section 14.2 previewed the false-positive argument: for $x \notin A$ the query compares two independent uniform r-bit strings, matching with probability 2^{-r}. This section makes the argument careful. The setup is small, the argument is two cases, the bound is $\varepsilon \leq 2^{-r}$, and the only nontrivial input is the uniformity of the fingerprint hash g, which the universal-hashing literature supplies.

Setup

Fix $A \subseteq U$ with $|A| = n$, a perfect hash $\phi : A \to [n]$ extended to all of U, and a fingerprint hash $g : U \to \{0,1\}^r$ with $r = \lceil -\log_2 \varepsilon \rceil$. The table is populated by (14.2): $F[\phi(a)] = g(a)$ for every $a \in A$. For a query point $x \notin A$,

$$\mathrm{P}\left(\textsc{Query}(x) = \top | x \notin A\right) = ?$$

where the probability is over the choice of g (ϕ is fixed by the build phase). The query reads slot $\phi(x)$, computes $g(x)$, and returns $\top$ iff $F[\phi(x)] = g(x)$; the question is when that holds for a non-member.

Two cases for $\phi(x)$

ϕ is injective on A but defined on all of U by its extension rule. For $x \notin A$, two situations arise.

Case (a): $\phi(x) \in \phi(A)$. The slot $\phi(x) = i$ coincides with $\phi(a^*)$ for the unique $a^* \in A$ mapped there, so $F[i] = g(a^*)$ and the query returns $\top$ iff $g(a^*) = g(x)$. Since $x \neq a^*$, uniformity of g makes $g(a^*)$ and $g(x)$ independent uniform on $\{0,1\}^r$, colliding with probability 2^{-r}.

Case (b): $\phi(x) \notin \phi(A)$. The extension may flag x as out-of-range, returning $\perp$ directly (no false positive), or map x to an unused slot $i \in [n] \setminus \phi(A)$. An unused slot holds either an explicit sentinel designed to mismatch every fingerprint (collision probability zero) or a residual value independent of $g(x)$

(collision probability at most 2^{-r}). In the sentinel variant, which the BHF assumes by default, case (b) contributes nothing.

The worst case is every $x \notin A$ falling into case (a), giving $\varepsilon = 2^{-r}$ exactly; the best case flags case (b) out-of-range, removing the non-image contribution. Both bracket the bound.

The bound

Proposition 14.4.1 (BHF false-positive rate)**.** *Let the BHF of Definition 14.2.1 be constructed for A with parameters ϕ, g, and $r = \lceil -\log_2 \varepsilon \rceil$, and assume g is uniformly distributed on $\{0,1\}^r$ over the query point. For every $x \notin A$,*

$$\varepsilon_{BHF} \;=\; \mathrm{P}\left(\textsc{Query}(x) = \top \middle| x \notin A\right) \;\leq\; 2^{-r} \;\leq\; \varepsilon.$$

The BHF meets the target false-positive rate by construction.

The proposition is the second half of the optimality story: with Theorem 14.3.1 it certifies the BHF as both space-optimal (minimum bits per element up to rounding) and accuracy-correct (delivers the target FPR). The inequality is tight in the sentinel-free variant (case (b) at arbitrary residual contents achieves 2^{-r} for every non-member) and strict in the sentinel variant (case (b) contributes zero, so the FPR is 2^{-r} weighted by the probability $\phi(x) \in \phi(A)$, strictly below 1 when U extends beyond A). The chained $2^{-r} \leq \varepsilon$ is a direct consequence of $r = \lceil -\log_2 \varepsilon \rceil$: ceiling rounding forces $r \geq -\log_2 \varepsilon$, hence $2^{-r} \leq \varepsilon$, with the over-achievement the same rounding slack Section 14.3 measured on the space side. One integer bit of width buys one integer bit of FPR precision.

The uniformity hypothesis

The hypothesis that g is uniform over the query point deserves comment, because it can fail adversarially: an attacker who knows g can choose x with $g(x) = g(a^*)$ for a target a^*, driving the per-query FPR to 1. Uniformity protects the average case, where query points are drawn independently of the implementation's choice of g, which covers most data-structure use.

The requirement is exactly pairwise independence of g. The collision probability $\mathrm{P}\left(g(a^*) = g(x)\right) = 2^{-r}$ holds whenever $g(a^*)$ and $g(x)$ are pairwise independent, and *universal hashing* in the sense of Carter and Wegman [CW79] delivers a family in which, for every $a^* \neq x$, $\mathrm{P}\left(g(a^*) = g(x)\right) \leq 1/|\text{codomain}|$ over the random choice of g. The BHF selects g from such

a family at build time and fixes it; the bound then applies pointwise over non-adversarial queries. In practice g is a wide cryptographic hash (SHA-256, BLAKE3) truncated to r bits or a high-quality non-cryptographic hash (MurmurHash3, xxHash), and the adversarial case is handled by salting g with a build-time secret, denying the attacker advance knowledge of the hash. Chapter 16 returns to the adversarial setting; here the non-adversarial hypothesis is the operating assumption.

Worked check at $r = 7$

The worked example of Section 14.2 took $\varepsilon = 0.01$, $r = 7$. The proposition predicts $\varepsilon_{\text{BHF}} \leq 2^{-7} = 1/128 \approx 0.0078$, below the target by a factor $0.0078/0.01 \approx 0.78$, so every non-member query has at most a 0.78% chance of a false positive against the 1% target. The 22% over-achievement is the rounding slack, and it matches on both sides of the trade-off: $1 - 2^{-r}/\varepsilon = 1 - 2^{6.644-7} = 1 - 2^{-0.356} \approx 0.219$, the same rounding that costs 0.356 bits per element on the space side. At a power-of-two target the slack vanishes: $\varepsilon = 2^{-7}$ is met exactly with $r = 7$. Awkward targets pay at most one bit per element and at most a factor of 2 in FPR, small against the Bloom filter's multiplicative $\log_2 e \approx 1.44$.

The check closes the loop. The construction satisfies the FNR-zero guarantee unconditionally and the FPR target under the uniformity hypothesis, at a budget of $\lceil -\log_2 \varepsilon \rceil$ plus the sublinear perfect-hash term. Section 14.5 turns to build time, query latency, and the static-versus-streaming trade-off that distinguishes the BHF from the Bloom filter in deployment.

14.5 Construction Time and Trade-offs

The space and FPR analyses settled the BHF's accuracy and bit budget. The remaining dimensions are build time, query latency, and the property that distinguishes the BHF from the Bloom filter in deployment: the BHF is static, so inserting a new element after build requires recomputing the perfect hash. This section quantifies the costs and offers a decision rule.

Building the perfect hash

The perfect hash $\phi : A \to [n]$ is built from A at build time, and several families achieve expected $O(n)$ construction. The classical Fredman-Komlos-Szemeredi scheme (1984) is $O(n)$ expected with $O(n \log n)$ output bits; Hagerup-Tholey (2001) reduces output to $O(n)$ bits; the displacement-based

family (MWHC, BBHash) is $O(n)$ expected with $O(n)$ bits at 2 to 4 bits per element; the BDZ construction (Belazzougui-Botelho-Dietzfelbinger, 2009) is $O(n)$ expected with the $O(n \log\log\log n / \log n)$ output that Section 14.3 used. The bibliographic entries appear in Section 14.6.

The expected-time guarantee comes from a randomization step: each algorithm samples a parameter (seed, displacement vector, partitioning hash) and retries on failure, with a constant expected number of retries. The construction dominates BHF build time, since the fingerprint fill is a single memory-bandwidth-bound pass. Modern displacement-based implementations sustain 10 to 50 million elements per second on commodity hardware, so build time is seconds for million-element sets and minutes for billion-element sets. Both the BHF and the Bloom filter scale linearly in n; the Bloom filter's per-element constant is smaller (a few hashes and bit-writes, no retries), so its absolute build time is lower at the same n. The BHF pays the perfect-hash construction up front in exchange for the per-element space saving.

Building the fingerprint table

Once ϕ is in hand, the table is filled in one pass over A: $F[\phi(a)] \coloneqq g(a)$, evaluating $\phi(a)$ and $g(a)$ once each and writing one r-bit slot per element, for $O(n)$ time and $O(nr)$ bits. The fill is embarrassingly parallel, since ϕ's bijectivity makes each iteration write a distinct slot, but it must follow the perfect-hash stage, which addresses the slots. For most builds the two-stage pipeline (perfect hash, then fingerprints) is the right organization, parallel within each stage.

Query time

The query of Algorithm 6 runs in $O(1)$: one perfect-hash evaluation, one fingerprint-hash evaluation, one memory read, and an r-bit comparison (constant when r fits a machine word, as it does for all practical targets). The Bloom filter's query costs k hash evaluations and k reads, with k at the optimum equal to

$$k^* \;=\; (m/n)\ln 2 \;=\; \log_2(1/\varepsilon),$$

since $m/n = \log_2(1/\varepsilon) \cdot \log_2 e$ (Proposition 8.4.1) and $\log_2 e \cdot \ln 2 = 1$; equivalently $\varepsilon = 2^{-k^*}$ at the half-occupancy optimum. The BHF's two hash evaluations therefore undercut the Bloom filter by a factor $k^*/2$. At $\varepsilon = 0.01$, $k^* = 6.64$, so the BHF performs about 3.3 times fewer hash evaluations per

query; at $\varepsilon = 10^{-6}$, $k^* = 19.93$, about 10 times fewer. The BHF is denser per bit and faster per query; the price for both is the static-set restriction.

Trade-offs versus the Bloom filter

On accuracy and space at the same target, the BHF dominates: it reaches the floor (Theorem 14.3.1), beating the Bloom filter's $\log_2 e$ overhead, and meets the FPR target (Proposition 14.4.1) under the same uniformity assumption. On cost-of-update the Bloom filter dominates: it inserts in $O(k)$ with no rebuild, while the BHF must rebuild the perfect hash, at $O(n)$, whenever A changes. Insertion of $a' \notin A$ is $O(k)$ for Bloom and $O(n)$ for the BHF (rebuild ϕ on $A \cup \{a'\}$, refill); deletion is unsupported by the standard Bloom filter (unsetting shared bits would create false negatives) and $O(n)$ for the BHF. Neither supports cheap deletion; the BHF supports expensive deletion, the Bloom filter none.

Table 14.2 summarizes.

Table 14.2: BHF versus standard Bloom filter at the same target ε and n. Space and query exclude the sublinear perfect-hash overhead.

	BHF	Bloom
Bits per element	$\lceil -\log_2 \varepsilon \rceil$	$\log_2(1/\varepsilon) \cdot \log_2 e$
Build time (whole set)	$O(n)$ expected	$O(kn) = O(n \log_2(1/\varepsilon))$
Insertion (post-build)	$O(n)$ (full rebuild)	$O(k) = O(\log_2(1/\varepsilon))$
Deletion	$O(n)$ (full rebuild)	not supported
Query	$O(1)$ (2 hashes, 1 read)	$O(k)$ (k hashes, k reads)
False-negative rate	0	0
False-positive rate	$\leq 2^{-r} \leq \varepsilon$	ε (asymptotic)

The trade-off is batch against stream: the BHF wins when the set is built once and queried many times, the Bloom filter when the set evolves and insertions are frequent relative to queries. A further structural cost of the BHF's data-dependent addressing deserves a remark, because it has no analogue in the Bloom filter.

Remark (BHFs do not merge by bit-OR)**.** *Two Bloom filters on disjoint sets A_1, A_2 with the same parameters merge into a filter on $A_1 \cup A_2$ by bitwise OR of their arrays, because the Bloom address space is data-independent: bit i has the same meaning in every instance. Two BHFs do not. The perfect hashes ϕ_1 and ϕ_2 are bijections onto overlapping ranges $[|A_1|]$ and $[|A_2|]$ with no shared semantics, so slot i in the two tables holds the fingerprints*

of unrelated elements; any pointwise combination produces an artifact corresponding to no valid BHF on the union. Merging requires rebuilding a fresh ϕ' on $A_1 \cup A_2$ and refilling, at $O(|A_1| + |A_2|)$, the same cost as building from raw input. The data-dependent addressing that gives the BHF its space optimality is exactly what forbids the cheap merge. △

Decision rule

The choice resolves on two questions: static or streaming set, and is space the binding constraint? Three combinations cover the practical cases.

Static set, space-critical: BHF. When the set is fixed at build time (or rebuilt on a schedule) and memory or transmission is the constraint, the $\log_2 e \approx 1.44$ saving translates directly to a smaller footprint. Examples: a membership filter shipped to clients on connection; a read-only index loaded into RAM at startup and queried millions of times; a content-delivery cache populated nightly and queried through the day.

Streaming set, space-tolerant: Bloom. When elements arrive continuously and must be absorbed in real time, and memory is plentiful, the 44% overhead buys $O(k)$ amortized insertion against $O(n)$ rebuild. Examples: a deduplication filter for ingest pipelines; an in-memory cache of recently-seen request IDs; a line-rate routing check.

Hybrid: tiered. When the set has a large static portion and a small streaming portion, the common case, a tiered structure keeps a static BHF over the base set and a small Bloom filter over the streaming delta, querying both ($\top$ if either reports $\top$) and absorbing the delta into a rebuilt BHF when it grows past a threshold of the base size. The hybrid gets most of the BHF's space efficiency on the static portion while accepting streaming updates through the delta, the merge cost being a BHF rebuild on $|A| + |\Delta|$ amortized over the inter-merge interval.

The BHF is not a universal replacement for the Bloom filter; it is the optimal choice when the set is static and the bit budget matters, the headline result delivered in exchange for the static-set restriction. Section 14.6 bridges to chapter 15, which builds the FPR-zero counterpart on the other half of the perfect-filter line.

14.6 Bridge

The chapter answered the optimality question chapter 8 left open. The Bernoulli Hash Function (Definition 14.2.1) reaches the entropy floor of

Theorem 8.3.1 up to integer rounding and a sublinear correction (Theorem 14.3.1), meets the FPR target under a standard uniformity assumption (Proposition 14.4.1), and pays for the improvement in static-set restrictions and an $O(n)$ build (Section 14.5). The $\log_2 e \approx 1.44$ factor the Bloom filter spends over the floor (Proposition 8.4.1) collapses to a constant additive overhead. The construction draws on a decades-old perfect-hash literature and the universal-hashing machinery the Bloom filter also uses, differing only in where the hash is applied; Section 14.6 names the antecedents.

The BHF occupies one end of the perfect-filter line of Definition 6.4.1, the FNR-zero membership structures. The other end holds the FPR-zero structures, the perfect hash filters of chapter 15. The two share a skeleton, a perfect hash addressing a per-element table, and trade their error modes: the BHF accepts a small false-positive rate to guarantee no false negatives; the BPHF (Bernoulli Perfect Hash Filter) guarantees no false positives at the cost of a controlled false-negative rate. The symmetry of the picture is not a symmetry of cost. A false-positive-free report is a coincidence made rare by one fingerprint slot, which the BHF buys cheaply; a no-false-positive guarantee is a certificate enforced against every non-member at once, and certifying a designated subset exactly is the expensive direction. Chapter 15 confronts that asymmetry rather than assuming it away. The applications split accordingly: the BHF where false negatives are unacceptable (a cache that must not miss a stored element, a routing layer that must not drop a known address), the BPHF where false positives are unacceptable (a precision-oriented index that must not return non-matching results, an authorization filter that must not admit an unauthorized identifier).

The BPHF reuses that skeleton, the same perfect hash and a per-slot table built in one pass, but its analysis is not a mechanical error-mode swap. The false-positive rate falls to zero only in a restricted query model, where non-members cannot present a valid witness (ranked search is the standard instance), and the per-element cost is $\lceil \log_2 n \rceil$ bits for a rank table rather than the BHF's $\lceil -\log_2 \varepsilon \rceil$ for a fingerprint table. A controlled false-negative rate is then bought by relaxing the perfect hash to a k-perfect one, $\omega \approx 1 - m/n$ at m slots, not by a fingerprint width. Chapter 15 develops the construction and the ranked-search application on the FPR-zero half of the line.

Bibliographic Notes

The BHF construction. The construction of Definition 14.2.1 and the optimality result of Theorem 14.3.1 follow Towell [Tow26m], "The Bernoulli Hash Function: Optimal Bernoulli Sets and Maps," which introduced the BHF as the constructive complement to the chapter-8 floor. The manuscript works in greater generality, extending the membership construction of Section 14.2 to approximate maps (each slot storing a key-value pair rather than a fingerprint); the space and FPR analyses here specialize that framework to set membership. A reader wanting the algebraic generalization, with the BHF as an instance of the optimal Bernoulli map, should treat the manuscript as the next reading. The perfect hash filter of chapter 15 is a distinct, complementary construction of the same set and map ADT, not an instance of the BHF: it builds by off-the-shelf perfect hashing at a constant $\log_2 e$ above the floor the BHF reaches by salt search.

The universal-hashing foundation. The fingerprint hash g of Proposition 14.4.1 inherits its uniformity from the universal hashing of Carter and Wegman [CW79], "Universal Classes of Hash Functions," the first family with provable pairwise independence (now 2-universal), bounding collision probabilities to $1/|\text{codomain}|$ over the family's random selection. The BHF needs only pairwise independence, so the later k-universal refinements (Wegman-Carter 1981, Siegel 1989) are not required, and Carter-Wegman remains the standard reference for the result the analysis depends on.

The modern perfect hash. The sublinear bound of Section 14.3 and the linear-expected-time algorithms of Section 14.5 trace to a long line culminating in Belazzougui, Botelho, and Dietzfelbinger [BBD09], "Hash, Displace, and Compress" (the BDZ scheme), which achieves $O(n)$ expected construction with output in $O(n \log\log\log n / \log n)$ at under 3 bits per element for n in the millions, combining hash-and-displace placement with entropy-efficient compression of the displacements. The chapter quotes its asymptotic bound as a black box. The antecedents BDZ improves on are Fredman-Komlos-Szemeredi (1984, the classical $O(n \log n)$-bit bound), Hagerup-Tholey (2001, the first $O(n)$-bit construction), and the MWHC and BBHash displacement families.

Comparison structures. Two contemporary designs occupy spaces adjacent to the BHF. Fan et al. [Fan+14]'s cuckoo filter stores fingerprints in

a cuckoo-hashed, cache-friendly layout, reaching about 1.05 to 1.3 times the floor (against the Bloom filter's 1.44) while supporting amortized $O(1)$ insertion and deletion; it recovers about half the Bloom overhead while remaining streamable. Bender et al. [Ben+12]'s quotient filter stores fingerprints in a quotient-style addressing scheme with similar trade-offs and streaming properties. Neither matches the BHF's asymptotic optimality, but both narrow the Bloom gap and extend the streaming case beyond what the BHF supports, making them the practical middle ground when a deployment needs both space efficiency and streaming updates.

Chapter 15

Perfect Hash Filters (BPHF)

15.1 From BHF to BPHF

Chapter 14 settled the FNR-zero arm of the perfect-filter line. This chapter turns to the FPR-zero arm. The same architectural template, a perfect hash addressing a per-element table, produces a structure with the dual error behavior: every report of membership is correct, and the only failure mode is silent omission. The two chapters together give the constructive realization of both arms of Definition 6.4.1, and neither arm is rhetorical or structurally easier than the other, as the asymmetry this chapter develops will show.

Recall the BHF

The Bernoulli Hash Function of Definition 14.2.1 sits on the FNR-zero arm: every $a \in A$ is reported as a member with certainty, and the only error is a non-member accepted by mistake. It addresses a fingerprint table $F : [n] \to \{0,1\}^r$ through a perfect hash $\phi : A \to [n]$ and compares the stored fingerprint $F[\phi(x)]$ to $g(x)$. At width $r = \lceil -\log_2 \varepsilon \rceil$ it buys $\varepsilon \leq 2^{-r}$ (Proposition 14.4.1) for r bits per element plus a sublinear perfect-hash correction, reaching the entropy floor of Theorem 8.3.1 up to integer rounding (Theorem 14.3.1) where the standard Bloom filter of Definition 5.2.1 spends $\log_2 e \approx 1.4427$ times the floor. Chapter 14 closed with the construction in hand and one question deferred: what happens on the other axis?

The opposite axis

The FNR-zero arm is the home for applications that cannot tolerate a missed member: a routing table that must never drop a known destination, a security check that must never admit a denied identifier, a cache that must never miss a stored key. A second class inverts the trade-off. Top-k retrieval must never surface an off-topic document as if it belonged in the top set, but can tolerate occasionally missing one that should have been included. A precision-oriented search index must never claim a non-matching record as a match, though the system can recover from an undercount by relaxing the query. A deduplication layer must never declare two distinct items the same, but a few surviving duplicates are recoverable by a later pass. In each, a false positive corrupts the output irrecoverably and a false negative is a tolerable miss the surrounding system absorbs. The desired guarantee flips: $\varepsilon = 0$ exactly, with $\omega > 0$ the controllable cost. Proposition 6.4.2 already established that the FPR-zero arm is closed under union and intersection, mirroring the FNR-zero arm; the algebraic shape is symmetric. What the present chapter supplies is the constructive question that shape leaves open: which structure achieves $\varepsilon = 0$ at a controllable ω, and at what space cost.

Construction sketch

The construction reuses the perfect hash $\phi : A \to [n]$ from chapter 14 unchanged and substitutes the second ingredient. In place of a fingerprint table storing $g(a)$, the BPHF stores a *rank value* $\rho(a)$ at slot $\phi(a)$, an integer in $[0, n)$ encoding a's position in some build-time ordering of A, derived from a secondary hash or from external metadata such as a relevance score. The rank table $R : [n] \to \{0,1\}^r$ uses $r = \lceil \log_2 n \rceil$ bits per slot, enough to distinguish the n ranks. A query computes $\phi(x)$, reads $R[\phi(x)]$, and returns $\top$ when the stored value equals $\rho(x)$.

The substitution looks like a free relabeling, and Section 15.2 shows it is not. Taken at face value over the whole universe, the rank comparison reproduces the BHF rather than flipping its axis: a non-member still collides with the stored rank with probability 2^{-r}. The $\varepsilon = 0$ guarantee appears only in a restricted query model, where ρ is a verifiable witness that genuine members carry and non-members cannot present. Ranked search is the standard instance of that model, and a controlled ω is then bought not by the rank comparison but by relaxing the perfect hash itself. The next section makes each claim precise and locates the asymmetry between the two arms of the line.

15.2 The BPHF Construction

Section 15.1 set the goal: a structure on the FPR-zero axis, mirroring the way the Bernoulli Hash Function sits on the FNR-zero axis. The plan keeps the perfect hash $\phi : A \to [n]$ and replaces the fingerprint table with a rank table the query verifies against the candidate's own rank. The construction is short; the honest accounting of what it does and does not deliver is the heart of the chapter, because the rank table alone does not flip the error axis, and seeing why reveals the asymmetry between the two arms of the perfect-filter line.

The two ingredients

Fix a finite universe U and a designated set $A \subseteq U$ with $|A| = n$.

(i) The perfect hash ϕ. The same perfect hash

$$\phi \ : \ A \ \longrightarrow \ [n], \qquad [n] = \{1, 2, \ldots, n\},$$

chapter 14 used: an injection on A, hence a bijection onto the addressing range, constructed at build time from A and extended to all of U. Its build and storage costs are identical to chapter 14's (Section 15.3). The BPHF reuses it without modification.

(ii) The rank table R and assignment ρ. A *rank table* $R : [n] \to \{0,1\}^r$ of n slots, each of width

$$r \ = \ \lceil \log_2 n \rceil, \tag{15.1}$$

the smallest integer with $2^r \geq n$. It is populated by a *rank assignment* $\rho : A \to \{0, 1, \ldots, n-1\}$ giving each member a distinct integer, from a position in a sorted ordering, a relevance score discretized to n buckets, or external metadata. Distinctness of the ranks is what fixes r at $\lceil \log_2 n \rceil$: it takes that many bits to label n elements apart.

Where the BHF compared a stored fingerprint $g(a)$ against the candidate's $g(x)$, the BPHF compares a stored rank $\rho(a)$ against the candidate's $\rho(x)$. The substitution is one line of pseudocode. Whether it changes the error analysis is the question the rest of the section answers, and the answer is more subtle than the symmetry of the picture suggests.

Construction and query

Given A, ϕ, and ρ, the table is built in one pass:

$$\text{for each } a \in A : \quad R[\phi(a)] \ := \ \rho(a), \tag{15.2}$$

writing each slot exactly once by bijectivity of ϕ, so the table is fully populated.

Definition 15.2.1 (Bernoulli Perfect Hash Filter)**.** *The* Bernoulli Perfect Hash Filter *(BPHF) for a designated set A with parameters ϕ, ρ, and $r = \lceil \log_2 n \rceil$ is the pair (ϕ, R) together with the membership query*

$$\textsc{Query}(x) \;=\; \top \;\; \textit{iff} \;\; R[\phi(x)] = \rho(x).$$

After the build pass (15.2)*, $x \mapsto \textsc{Query}(x)$ is the membership report of the approximate set $\mathsf{A}^{\pm}$.*

The definition has the shape of Definition 14.2.1: a perfect-hash addressing function, a per-slot value, and a query that reads one slot and compares it to a computed value. The only change is which value is stored, a uniform fingerprint g for the BHF, an application-chosen rank ρ for the BPHF. Algorithm 7 writes it out.

Algorithm 7: BPHF construction and query

```
1  procedure BUILD(A, ρ)
2  |  construct perfect hash φ : A → [n];
3  |  allocate rank table R of size n, each slot r bits;
4  |  for each a ∈ A do
5  |  |  R[φ(a)] ← ρ(a);
6  |  end
7  |  return (φ, R);
8  function QUERY(x)
9  |  i ← φ(x);
10 |  v ← ρ(x);
11 |  if R[i] = v then return ⊤;
12 |  return ⊥;
```

Against the BHF pseudocode of Algorithm 6, the only changed line substitutes ρ for g in the value compared. The two algorithms share their performance profile. Whether they share their error profile is the question, and it needs care.

What the bare construction delivers

Read the algorithm directly. For a member $a \in A$, bijectivity sends a to a slot written by a alone, where the build stored $\rho(a)$; the member recomputes

$\rho(a)$, reads its own slot, and matches. The match always succeeds, so $\omega = 0$. Two members with the same rank do not interfere, because ϕ sends them to different slots and each reads back the rank it wrote; the rank-collision worry the symmetry argument might raise does not arise.

For a non-member $x \notin A$, the extended hash sends x to a slot $\phi(x) = \phi(a^*)$ owned by some member a^*, holding $\rho(a^*)$. The query accepts x exactly when the recomputed $\rho(x)$ equals $\rho(a^*)$. If ρ behaves like a uniform r-bit hash, the two agree with probability 2^{-r}, so $\varepsilon = 2^{-r}$, exactly the fingerprint-collision probability Proposition 14.4.1 bounded. The bare BPHF, queried over the full universe with a recomputable rank, has the same error profile as a BHF at the same width: zero false negatives, false positives at 2^{-r}. Renaming a fingerprint a rank changes nothing the query can observe.

Its rates match a BHF's, but as a *construction* this bare structure is not the BHF. A perfect hash with a per-slot fingerprint of width $r = \lceil -\log_2 \varepsilon \rceil$ is the *perfect hash filter*, the FNR-zero positive Bernoulli set at false-positive rate $\varepsilon = 2^{-r}$, and it is the *practical* construction of the set abstract data type, as against the space-optimal BHF of chapter 14. The two are distinct, complementary constructions of the same object, not one an instance of the other. The BHF reaches the information-theoretic floor $-\log_2 \varepsilon$ exactly, but builds by searching for a salt under which an acceptance predicate holds; the perfect hash filter pays a constant $\log_2 e \approx 1.44$ bits per element above that floor, and in return builds by off-the-shelf minimal perfect hashing, which is fast and widely deployed. That overhead is a vanishing fraction of the optimum as $\varepsilon \to 0$, so the perfect hash filter is succinct in the relative sense while conceding the absolute optimum to the BHF. The rank table of Definition 15.2.1 is this same construction read on the FPR-zero axis: it fixes $r = \lceil \log_2 n \rceil$ so the slot can carry a distinct rank, and the witness model of the next subsection turns the result into an exact filter.

This is the asymmetry Section 15.1 promised. A false positive is a coincidence, one stored value colliding with one recomputed value, and a single r-bit slot makes it rare; a false-negative-free filter is therefore cheap, bought for r bits per element. The FPR-zero axis asks the opposite, that no non-member is ever accepted, and that is not a coincidence to make rare but a certificate to enforce against every one of the $|U| - n$ non-members at once. Certifying a subset exactly is the expensive direction. The next two subsections give the two honest routes to it.

How the false-positive rate becomes zero

The bare construction admits a non-member only when its recomputed rank equals the rank stored at the slot it lands in. There is exactly one way to close that gap without storing a universe's worth of information: arrange the query so a non-member never presents a matching rank in the first place. Call ρ a *witness* for A when the recomputed $\rho(x)$ equals the stored rank at slot $\phi(x)$ if and only if x is the member that claimed that slot. Under a witness, $R[\phi(x)] = \rho(x)$ holds only for members, so

$$\varepsilon \;=\; 0$$

holds exactly, by construction rather than in expectation. The relevance ranking of a search corpus is the standard example, worked in Section 15.4: a query is posed only for a document that already carries its corpus rank, an outside document has no such rank to present, and the witness condition is met.

The witness model is not a free upgrade. It restricts the query domain to candidates that arrive with a verifiable rank, and it pays for the guarantee in the only currency that buys it: the rank table stores, in n slots of $\lceil \log_2 n \rceil$ bits, enough to name every member apart and certify a presented witness against the right slot. That is the cost of certifying a subset exactly, the same counting argument that set the book's earlier space bounds. A structure that accepts no non-member must in effect encode which elements it accepts, and encoding a designated subset of an arbitrary universe costs on the order of $\log_2 n$ bits per element once the query domain is the corpus itself. A genuinely universe-wide FPR-zero filter, rejecting every outsider by recomputation alone, would store on the order of $\log_2 |U|$ bits per element, and at that price it is a dictionary, not a compact filter.

The witness model is a property of the query, not of the table, and the distinction has teeth.

Remark (The FPR-zero guarantee lives in the query model)**.** *The $\varepsilon = 0$ guarantee holds only when ρ is a witness that members possess and non-members cannot forge; it is not a property of the stored table. A degenerate assignment, $\rho(a) = 0$ for all a, writes zero into every slot, keeps $\omega = 0$, and admits every non-member whose presented rank is zero, so the false-positive rate is the fraction of queried non-members presenting zero. A low-entropy public ρ an adversary can evaluate on any candidate, such as a coarse document-length bucket, is similarly forgeable: a non-member presents a rank matching its slot with non-negligible probability. The guarantee survives only for a ρ that off-corpus candidates cannot recompute, a*

corpus sort position or a salted hash of corpus-only metadata. Chapter 16 returns to the adversarial query in the broader catalogue, and the keyed-witness construction is the bridge to the cipher maps of Part V. △

With distinct ranks and a true perfect hash, the witness model places the BPHF at the *corner* of the perfect-filter line: both error rates are zero on the query domain. The structure is exact there, and exactly for that reason not yet an approximate set. The mirror of the BHF, a tunable structure trading a controlled error for space, comes from relaxing the perfect hash.

A controlled false-negative rate via k-perfect hashing

The BHF moves along its axis by changing the fingerprint width. The BPHF needs the matching dial on its own axis, a way to spend less space for a controlled, nonzero ω while holding $\varepsilon = 0$. The dial is the perfect hash. A *k-perfect hash* relaxes injectivity, allowing up to k members to share a slot. It is cheaper to find and uses fewer slots, but a shared slot holds only one member's rank; the members that lose the contention read a different member's rank, fail the comparison, and are reported as non-members. These lost members are the only error the k-perfect BPHF makes, and a non-member still cannot present a witness, so ε stays exactly zero. With m slots holding n members under a balanced construction, roughly $n - m$ are crowded out, so

$$\omega \;\approx\; 1 - \frac{m}{n}, \tag{15.3}$$

zero at $m = n$ (the exact corner) and growing as the table shrinks. A table at ninety percent of full size misses about one member in ten while admitting no non-member at all. This is the mirror of the BHF made precise: the BHF holds $\omega = 0$ and dials ε down by adding fingerprint bits; the BPHF holds $\varepsilon = 0$ and dials ω down by adding slots. Each lives on its own axis of Definition 6.4.1, paying for accuracy in space by Proposition 6.4.2.

A worked example

Fix $n = 1000$. The rank width is

$$r \;=\; \lceil \log_2 1000 \rceil \;=\; \lceil 9.966 \rceil \;=\; 10 \text{ bits},$$

since $2^{10} = 1024 \geq 1000 > 2^9 = 512$. The rank table occupies $nr = 10{,}000$ bits (1250 bytes), plus a perfect-hash representation of a few bits per element by the chapter 14 analysis, for a total near 12,000 to 13,000 bits.

At the corner, a true perfect hash with distinct ranks in the witness model, $\varepsilon = \omega = 0$: the structure never accepts a non-corpus candidate and never misses a corpus member, a perfect-hash dictionary viewed as the corner of the line. To make it a tunable approximate set, shrink to a k-perfect hash with $m = 900$ slots. By (15.3),

$$\omega \;\approx\; 1 - \tfrac{900}{1000} \;=\; 0.10, \qquad \varepsilon \;=\; 0,$$

so about one corpus member in ten is missed while no outside candidate is ever accepted. For comparison, a BHF on the same set at $\varepsilon = 0.01$ uses $r = 7$ bits per element (7000 bits) and gives $\omega = 0$ at $\varepsilon \leq 0.0078$. The two answer opposite needs: the BHF never misses a member and rarely admits a stranger; the BPHF never admits a stranger and occasionally misses a member. An application that can tolerate neither must pay the dictionary price for both. Section 15.4 works through one setting where the BPHF's profile is exactly the one the problem wants.

15.3 Space and Time Analysis

The BPHF stores the perfect hash ϕ and the rank table R of width $r = \lceil \log_2 n \rceil$. This section totals the bits, gives the build and query times the construction inherits from its chapter 14 ancestor, and tabulates the trade against the BHF. The headline is Proposition 15.3.1: the per-element cost is $\lceil \log_2 n \rceil + o(1)$, with the perfect-hash overhead vanishing as in Section 14.3. The build is expected $O(n)$ and queries $O(1)$, matching the BHF on the time axes. The structural difference is on the space axis: the BPHF's bits-per-element depends on the set size n, the BHF's on the target rate ε, so the two families are not substitutes parameter-for-parameter.

Rank table storage

The table $R : [n] \to \{0,1\}^r$ holds n slots of r bits, so

$$L_R \;=\; n \cdot r \;=\; n \cdot \lceil \log_2 n \rceil \tag{15.4}$$

bits, growing as $n \log_2 n$, the same information-theoretic cost a sorted lookup index pays to distinguish n ranks. At $n = 10^6$ the table is $20 \cdot 10^6$ bits, about 2.5 megabytes; at $n = 10^9$ it is $30 \cdot 10^9$ bits, about 3.75 gigabytes. The integer rounding in $r = \lceil \log_2 n \rceil$ adds between 0 and 1 bits per element over $\log_2 n$, zero at powers of two and averaging 0.5, the rounding profile Section 14.3 showed for fingerprints; fractional-bit rank encodings are not

worth the query-time complexity, the same conclusion chapter 14 reached. The table is a flat array addressed by ϕ, with no padding, the densest representation of one rank per element.

Perfect-hash storage

The perfect hash is a property of the set, not of the values stored at each slot, so the BPHF inherits the BHF's overhead unchanged: the Belazzougui-Botelho-Dietzfelbinger family gives

$$L_\phi \;=\; O\big(n \log\log\log n / \log n\big)$$

bits, $o(n)$ and therefore vanishing relative to the rank table's $n \log_2 n$, at practical constants of 2 to 3 bits per element at $n = 10^6$ and below that at larger n. The chapter 14 build pass populated $F[\phi(a)] = g(a)$; the chapter 15 pass populates $R[\phi(a)] = \rho(a)$. Only the per-slot value differs; the perfect-hash representation, slot count, and addressing are identical, so the chapter does not redo the derivation.

Total bits per element

Combining,

$$L_{\mathrm{BPHF}} \;=\; L_R + L_\phi \;=\; n \cdot \lceil \log_2 n \rceil \;+\; O\big(n \log\log\log n / \log n\big), \qquad (15.5)$$

and dividing by n,

$$L_{\mathrm{BPHF}}/n \;=\; \lceil \log_2 n \rceil \;+\; O(\log\log\log n / \log n). \qquad (15.6)$$

Proposition 15.3.1 (BPHF space)**.** *The Bernoulli Perfect Hash Filter of Definition 15.2.1 on a designated set A with $|A| = n$ uses*

$$L_{BPHF}/n \;=\; \lceil \log_2 n \rceil \;+\; o(1) \;=\; \log_2 n \;+\; O(1)$$

bits per element, the rank table dominating at $\lceil \log_2 n \rceil$ bits and the perfect-hash overhead a vanishing sublinear correction.

The proposition has the form of Theorem 14.3.1 with one difference: the leading term is controlled by the set size n, not the target rate ε. The two budgets

$$L_{\mathrm{BHF}}/n = \lceil -\log_2 \varepsilon \rceil + o(1), \qquad L_{\mathrm{BPHF}}/n = \lceil \log_2 n \rceil + o(1)$$

are functions of different inputs. The BHF spends bits at a rate the designer sets through ε; the BPHF at a rate the set size dictates. Neither is a defect: each budget is the minimum its guarantee requires. The BHF pays $\lceil -\log_2 \varepsilon \rceil$ bits to make a false-positive coincidence rare; the BPHF pays $\lceil \log_2 n \rceil$ bits so the rank assignment can name all n members apart and certify a witness. At full resolution the BPHF is exact (the $\omega = 0$ corner); the k-perfect dial of Section 15.2 shrinks the slot count to $m < n$ and trades $\omega \approx 1 - m/n$ for the saved space.

Build and query time

The build writes one rank per element in a one-pass $O(n)$ stage plus the expected-$O(n)$ perfect-hash construction of Section 14.5 (the same Fredman-Komlos-Szemeredi, Hagerup-Tholey, MWHC, BBHash, and BDZ families), dominated by the latter; build throughputs and times match the BHF's. The rank-fill is embarrassingly parallel (distinct slots by bijectivity), running after the perfect-hash stage in the same two-stage pipeline. The only build-time difference from the BHF is computing a rank $\rho(a)$ rather than a fingerprint $g(a)$ per element, both $O(1)$.

The query of Algorithm 7 runs in $O(1)$: one perfect-hash evaluation, one rank evaluation, one memory read, and an r-bit comparison (at machine-word speed for any n up to 2^{64}). The cycle count is essentially the BHF's, differing only in which function is the second evaluation and what the table read returns.

The bit-cost picture at $n = 10^6$

Fix $n = 10^6$. The BPHF's per-element cost is $\lceil \log_2 10^6 \rceil = 20$ bits, independent of any error target; the BHF's is $\lceil -\log_2 \varepsilon \rceil$. Table 15.1 puts them side by side.

Table 15.1: Bits per element at $n = 10^6$. BHF column is $\lceil -\log_2 \varepsilon \rceil$ (Theorem 14.3.1); BPHF column is $\lceil \log_2 n \rceil = 20$ (Proposition 15.3.1), constant since the BPHF guarantees $\varepsilon = 0$. Both omit the sublinear perfect-hash overhead.

ε	BHF	BPHF
10^{-2}	7	20
10^{-3}	10	20
10^{-6}	20	20

The two filters cross over at $\varepsilon = 1/n$, where $\lceil -\log_2 \varepsilon \rceil = \lceil \log_2 n \rceil$: at $n = 10^6$ this is $\varepsilon = 2^{-20} \approx 10^{-6}$. Above the crossover (looser targets, $\varepsilon > 1/n$) the BHF is cheaper; below it (stricter targets) the BPHF is actually cheaper at the same n. The two regimes are the two operating modes, the BHF where the application tolerates some false positives for cheap storage, the BPHF where it cannot tolerate any. The BPHF's extra bits in the loose-target regime are not wasted: they buy a guarantee the BHF cannot deliver at any bit budget. A BHF at $\varepsilon = 10^{-2}$ admits roughly one false positive per hundred non-member queries no matter how many bits it spends; a BPHF at the same n admits zero. The 13 extra bits per element are the price of converting a 10^{-2} probability into a certainty on the consequential axis, the conversion Section 15.4 shows paying for itself in top-k retrieval.

15.4 Ranked Search Application

The chapter developed the BPHF abstractly: zero false positives in the witness model, exact at full resolution, a controlled $\omega \approx 1 - m/n$ bought by a k-perfect hash of m slots at $\lceil \log_2 n \rceil$ bits per element (Proposition 15.3.1). This section places it in its archetypal application, which also supplies the witness model the guarantee needs. The setting is ranked search: given a corpus and a query, return the top-k most relevant documents. The $\varepsilon = 0$ guarantee is exactly the precision invariant the application needs, and the BPHF is not a generic structure that happens to fit ranked search but the structure ranked search asks for.

The ranked-search problem

Fix a corpus of n documents with IDs $a_1, \ldots, a_n$ and a relevance score $s : \mathrm{ID} \to \mathbb{R}$ from term weights, embeddings, or metadata. A query asks for the top-k IDs by score. The exact solution scores every document and sorts, at $O(n)$ score evaluations plus $O(n \log n)$ sorting, dominated by the scoring when s is expensive. At corpus sizes in the millions the question becomes whether top-k retrieval can avoid scoring every document at every query. A structure that pre-computes the answer once and serves queries without per-document scoring is the architectural answer, and the BPHF is one.

BPHF for top-k retrieval

The build is a single pass over the corpus: pre-compute $s(a_i)$ for every document, sort the IDs by descending score, and assign each its rank, the

top-scoring ID rank 0 and the bottom rank $n-1$. That rank assignment $\rho : A \to \{0, \ldots, n-1\}$ is exactly the input Section 15.2 requires; build the BPHF on it with $r = \lceil \log_2 n \rceil$ bits per slot. Top-k retrieval reduces to a rank-bound check: a candidate a is in the top k iff $\rho(a) < k$, so

$$\textsc{TopK}(a) \;=\; \textsc{Query}(a) \;\wedge\; R[\phi(a)] < k,$$

combining the BPHF membership check with a numeric comparison. The query is $O(1)$: one perfect-hash evaluation, one rank evaluation, one memory read, one r-bit comparison, one integer comparison against k, constant in both n and k. The build is the one-time $O(n \log n)$ sort plus the $O(n)$ BPHF build of Section 15.3; each query then saves the n score evaluations the exact solution repeats, dropping per-query cost from $O(n)$ to $O(1)$. The profile is batch build and constant-time query, matching the BHF deployment of chapter 14: a search engine rebuilds on each hourly corpus snapshot and serves millions of queries per second between rebuilds; a recommender rebuilds per-user indices nightly and serves personalized top-k instantly.

Why $\varepsilon = 0$ is the right guarantee

Top-k retrieval has an asymmetric error structure. A false positive surfaces an off-topic document as if it belonged in the top set: the user sees an obvious irrelevance, clicks it, finds it useless, and judges the system broken. The cost is corrupted output, visible and attributable. A false negative omits a document that should have appeared: the user sees a top-100 list that is really a top-99-plus-one-missing, usually does not notice, and recovers the entry by relaxing the query when they do. The cost is mild under-recall, mostly invisible and recoverable. The asymmetry tracks the BPHF exactly: the $\varepsilon = 0$ guarantee of Definition 15.2.1 eliminates the high-cost error by construction, and the $\omega > 0$ trade admits the low-cost error at a controlled rate. The designer who chooses the BPHF makes the trade explicitly, zero corrupted output for a small under-recall, the right trade wherever precision is the binding metric and recall the recoverable one: ranked search, recommendation, precision-sensitive classification, authorization filters.

The false-negative cost

Ranked search is the witness model: a query is posed for a document that already carries its corpus rank, and a non-corpus candidate has none to present, so $\varepsilon = 0$. The false negatives come entirely from the k-perfect hash. At full resolution every document owns a slot and the retrieval is exact;

shrinking to $m < n$ slots crowds out a fraction $1 - m/n$ of the corpus, and by linearity over the k true top documents the expected misses are

$$\mathbb{E}[\text{misses}] \;\approx\; k \cdot \omega \;\approx\; k\left(1 - \frac{m}{n}\right), \tag{15.7}$$

using $\omega \approx 1 - m/n$ from (15.3), set by the retrieval depth k and the table ratio m/n and independent of the rank width $r = \lceil \log_2 n \rceil$.

Proposition 15.4.1 (BPHF top-k expected misses)**.** *Let the BPHF of Definition 15.2.1 be built on a corpus of n documents in the witness model, with a k-perfect hash of $m \le n$ slots. A top-k query surfaces no off-corpus document ($\varepsilon = 0$) and misses, in expectation, at most $k\,(1 - m/n)$ of the true top-k documents. At full resolution $m = n$ the retrieval is exact.*

Halving the spare capacity $1 - m/n$ halves the expected misses; $m = n$ removes them. The designer sets the table size to make the per-query miss expectation small enough that the experience is indistinguishable from exact top-k, paying in slots for recall.

Remark (Composing the two filter families)**.** *The two arms compose. To test $x \in A \cap B$ with a BPHF on A and a BHF on B, query both and return $\top$ iff both do, at $O(1)$. The composed false-positive rate is bounded by the BHF's alone: a false positive needs $x \notin A \cap B$ with both filters accepting, and if $x \notin A$ the BPHF rejects with certainty ($\varepsilon = 0$), so only the $x \in A$, $x \notin B$ case contributes, at the BHF's $\le 2^{-r_B}$. The composed false-negative rate is the BPHF's: for $x \in A \cap B$ the BHF accepts with certainty, so a miss requires the BPHF to miss, at 0 with a true perfect hash and $\approx 1 - m_A/|A|$ under a k-perfect one. The composition inherits the weaker guarantee on each axis, the FPR-zero arm absorbing the non-members of A before the BHF is consulted.* △

A worked example

Fix $n = 10^6$ documents and top-$k = 100$. The rank width is

$$r \;=\; \lceil \log_2 10^6 \rceil \;=\; \lceil 19.93 \rceil \;=\; 20 \text{ bits},$$

so the rank table is $2 \cdot 10^7$ bits, 2.5 megabytes, and with the perfect-hash representation about 2.7 to 2.9 megabytes, comfortably in a server's L3 cache. At full resolution $\varepsilon = \omega = 0$ and the retrieval is exact. Shrinking to $m = 0.99\,n$ slots, (15.7) gives

$$\mathbb{E}[\text{misses}] \;\approx\; 100\,(1 - 0.99) \;=\; 1,$$

so a typical top-100 query returns about 99 true top documents while $\varepsilon = 0$ holds exactly, not one of the roughly $|U| - n$ off-corpus candidates ever surfaced. The one-percent table saving costs about one missed result in a hundred, and the affirmative guarantee is untouched; the retrieval is indistinguishable from exact top-k, the missing entry rarely noticed and a wrong entry never present.

A full sorted index storing each rank explicitly costs the same $n \log_2 n$ bits for ranks but adds an associative structure mapping IDs to ranks, a hash table or balanced tree at 32 to 64 bits per document of pointers and load factor, with $O(1)$ amortized or $O(\log n)$ queries. The BPHF provides the same lookup at one memory access and one comparison, the minimum any structure can achieve, and folds the $\varepsilon = 0$ membership check into the rank lookup, where the sorted index would need a separate check to filter off-corpus queries. The construction also extends, preserving $\varepsilon = 0$ throughout: a multi-resolution variant stores more bits for high-ranked documents and fewer for the tail; a streaming-rebuild variant amortizes inserts over the inter-arrival interval; a tiered variant pairs a recent-corpus BPHF with an archived-corpus BPHF and merges their verdicts.

The application illustrates Part IV's larger point: the perfect-filter line of Definition 6.4.1 is an architectural decision space that maps to operational requirements. Precision-binding applications take the FPR-zero arm and the BPHF; recall-binding applications take the FNR-zero arm and the BHF. The decision is forced by which error the application can tolerate, and the chapter's two constructions cover both halves at the asymptotic bit-budget optimum on each side. Section 15.5 places them in Part IV's architecture and turns to Part V.

15.5 Bridge to Part V

The chapter assembled the constructive answer to the FPR-zero arm of the perfect-filter line. The BPHF of Definition 15.2.1 reaches $\varepsilon = 0$ in the witness query model, is exact at full resolution, and trades a controlled $\omega \approx 1 - m/n$ for a smaller table through a k-perfect hash, at $\lceil \log_2 n \rceil + o(1)$ bits per element (Proposition 15.3.1), applying cleanly to ranked retrieval through the construction of Section 15.4.

Closing Part IV

Part IV has a clean shape in retrospect. Chapters 5 and 6 named the perfect-filter line (Definition 6.4.1, building on Definition 5.4.1); chapter 8

derived the entropy floor (Theorem 8.3.1) as the bit-budget lower bound any structure on the line must respect; chapters 14 and 15 constructed the two operating points, each at one extreme error axis, each reaching its bound up to integer rounding and a sublinear correction. The BHF reaches the floor on the FNR-zero arm, dissolving the standard Bloom filter's $\log_2 e \approx 1.44$ overhead into a constant additive one (Theorem 14.3.1); the BPHF reaches the analogous bound on the FPR-zero arm, with the budget shifted from $\lceil -\log_2 \varepsilon \rceil$ to $\lceil \log_2 n \rceil$ and the controlled error from ε to ω. The two are the endpoints $(\varepsilon, 0)$ and $(0, \omega)$ of the line; a structure that tolerates both errors can interpolate between them, trading one axis against the other, but the endpoints are where the bit budget is minimized for a one-sided guarantee. The space-accuracy duality of the chapter 6 geometry becomes a deployment decision once the operating points are in hand: false-negative-intolerant applications take the BHF, false-positive- intolerant ones the BPHF.

Part V preview

Part V picks up two threads. The first is diagnostic: chapter 16 catalogues the structural ways a system fails to be Bernoulli, varying rates, correlated errors, and adversarial inputs that break the fixed-rate axioms, giving a checklist for when the framework applies cleanly and when it needs modification. The witness model's dependence on an unforgeable rank (Line 12) is one entry in that catalogue and the hinge to the second thread, which is generative: chapters 17 and 18 develop *cipher maps*, recasting the perfect-hash-and-fingerprint architecture as the building block for encrypted search, with the perfect hash replaced by a trapdoor whose forward direction is public and reverse private. The FPR and FNR guarantees of the present part carry into the cryptographic setting on the same architectural template. Section 15.5 attaches the construction to its antecedents, including the *maph* C++ library that is its production-grade realization.

Bibliographic Notes

The perfect hash filter and the BPHF. The bare construction of Section 15.2, a minimal perfect hash with a per-slot fingerprint, is the *perfect hash filter* of Towell [Tow26n], "The perfect hash filter," which develops it as the FNR-zero positive Bernoulli set and as the practical, off-the-shelf-constructible counterpart to the space-optimal BHF, lifts it to a key-value map, and applies it to rank-ordered retrieval. That manuscript and the

BHF manuscript [Tow26m] are explicit that the two are distinct, complementary constructions of the same set and map abstract data type, not one an instance of the other: the BHF reaches the space floor by salt search, the perfect hash filter pays $\log_2 e$ above it for a perfect-hash build.

The FPR-zero reading of the structure, the rank table of Definition 15.2.1, and the witness model (Line 12) that drives the false-positive rate to zero are this chapter's own development on the dual axis; they extend the manuscript's construction rather than appear in it. The ranked search of Section 15.4 is the witness-model instance of the rank-ordered retrieval the manuscript treats through its map generalization.

The perfect-hashing literature. The perfect hash ϕ inherits its construction and bounds from the literature surveyed in §14.7. The two foundational entries are Carter and Wegman [CW79], "Universal Classes of Hash Functions," which established the pairwise independence the rank-comparison check rests on, and Belazzougui, Botelho, and Dietzfelbinger [BBD09], "Hash, Displace, and Compress" (the BDZ scheme), which achieves $O(n)$ expected construction and $O(n \log\log\log n / \log n)$ output bits at under 3 bits per element for n in the millions. The BPHF inherits both unchanged, because the perfect-hash construction depends on A alone and not on what is stored at each slot, and the choice among the displacement-based families (with Fredman-Komlos-Szemeredi, Hagerup-Tholey, MWHC, and BBHash the historical line BDZ caps) is an implementation detail that does not affect the space or time bounds.

The ranked-search literature. The application of Section 15.4 sits in a long line of work on probabilistic information retrieval, the standard treatment being Manning, Raghavan, and Schütze [MRS08], *Introduction to Information Retrieval*. The line runs from the Robertson-Spärck Jones framework of the 1970s through TF-IDF weighting and BM25 to modern neural-embedding scoring. The chapter is agnostic about which scoring function s supplies the ranks; the BPHF accommodates any whose output reduces to a per-document rank at build time. Its contribution is not the scoring function but the post-scoring indexing structure for top-k serving.

The *maph* reference implementation. The perfect hash filter has a reference implementation, the *maph* library (C++), maintained in its own repository rather than bundled with the manuscript. It realizes the perfect-hash-plus-fingerprint construction over a configurable displacement-based

perfect-hash backend, and is the natural artifact against which to measure the per-element overhead the chapter reports against the Proposition 15.3.1 accounting. The FPR-zero rank-table variant of this chapter is not what maph implements; the measured comparison against the formula is left to a separate empirical study.

Part V

The Boundary and Beyond

Chapter 16

Approximate but Not Bernoulli

16.1 What This Chapter Is For

The preceding chapters wrote the affirmative theory of the Bernoulli framework and, at several points, set aside a phenomenon that does not fit the two axioms cleanly: a rate that drifts, a pair of queries that share a noise source, an input chosen by an adversary. Qualifying each result in place would have buried the positive content under footnotes; the negative space is collected here instead. This chapter maps it, naming for each failure the axiom it violates and the analytical tool that replaces the broken machinery, and Section 16.6 recovers the part of the framework that survives.

Three failure modes

The phenomena set aside earlier fall into three modes, each taken up in its own section.

Varying rates over time (Section 16.2). The error rate ε (or ω) is not a stable property of the channel. It drifts with the query index, because the structure changes (a Bloom filter filled past capacity), the environment changes (a sensor's noise grows with temperature), or the calibration data did not represent the deployment data (classifier drift). This violates Axiom 2 (Axiom 2): the per-block rates are no longer stable across queries. The channel matrix Q becomes a sequence (Q_t), and the iid concentration tools of chapter 11 give way to their martingale counterparts.

Correlated noise across queries (Section 16.3). Two queries to the

same structure share error sources: the same query repeated is deterministic, two queries to overlapping Bloom-filter bits are positively correlated, sequential draws from an online learner carry a temporal dependence. This violates Axiom 1 (Axiom 1): the error events at distinct inputs are no longer mutually independent. The factored joint distribution behind every multi-query formula becomes an inequality, and the composition theorem of chapter 3 picks up correction terms whose sign tracks the sign of the correlation.

Adversarial inputs (Section 16.4). The query stream is not iid; the queries are chosen by an adversary, or an adaptive learner, to expose the worst case. Both axioms can break: the adversary's t-th query is conditional on the responses to the earlier ones (so the inputs are not independent), and the worst-case query sits at the tail of whatever per-block rate distribution the channel admits (so the single ε the analyst computed no longer describes the queries that actually arrive). The right tools are worst-case bounds, cryptographic hash families, and trapdoor primitives, not concentration inequalities.

Posture

The chapter is written under a deliberate honesty discipline. Where the framework genuinely does not apply, it says so, names the violated axiom, and points to the replacement tool; it does not stretch the axioms to cover cases where they manifestly fail. Section 16.6 pushes back against pure relinquishment by cataloguing what survives: the perfect-filter line of chapter 6 still classifies systems by their rate-plane position when the rates are time-varying, the channel-matrix view of chapter 9 still applies query by query when the matrix is not stable, and the PPV/NPV machinery of chapter 7 still holds at any single query even when the prevalence drifts. These are partial successes, catalogued honestly rather than oversold.

16.2 Varying Rates Over Time

The first failure mode is the most common in practice. The error rate is not a fixed property of the channel; it changes with the query index. Where the earlier chapters wrote ε as a scalar, this section writes $\varepsilon(t)$ and asks what happens to the iid analysis when the scalar reading is no longer available.

The phenomenon

Three examples recur through the chapter. A *drifting classifier*, trained on one distribution and deployed on another, sees its per-query false-positive rate climb as the deployment distribution moves away from the training one, from $\varepsilon(t) \approx 0.02$ early to 0.15 months later. A *Bloom filter past capacity*, sized for $n = 1000$ at $\varepsilon \approx 0.01$ but loaded with 5000 items, watches its fill ratio rise from 30% toward 100% and its false-positive rate climb from 0.01 to about 0.5, following $\varepsilon(t) \approx (1 - e^{-kt/m})^k$ from Section 5.2 with t the insertion count at query time. A *drifting sensor* sees its error rate track its operating temperature over the day. What unites the three is structural: the rate is a non-constant function of the query index, and treating it as constant produces bounds that underestimate the actual error frequency, sometimes by large factors.

Which axiom is violated

Section 4.3 stated Axiom 2 in two equivalent forms, a conditional independence of the per-block rates and a stability of the channel matrix Q across queries. The stability form breaks here: the per-query rate at step t is $\varepsilon(t)$, not ε, so the joint distribution over a query sequence no longer factors through a single Q. The practical signature is that the same query, repeated at different times, draws different rate parameters. Time-varying rates are compatible with Axiom 1 (Axiom 1) in many cases: if the drift is governed by an exogenous time variable and errors at distinct queries remain conditionally independent given the drift state, only Axiom 2 fails. The two axioms are logically independent.

Consequences for the iid tools

Three workhorses break. The *composition theorem of chapter 3* ($\eta_{\text{total}} = 1 - \prod_i (1 - \eta_i)$) assumed each η_i fixed; when the η_i are time-indexed or random, the product is an average over a trajectory, and the error along the worst trajectory can exceed it. The *PPV formula of chapter 7* (PPV $= \pi\tau/(\pi\tau + (1-\pi)\varepsilon)$) still holds at any single query (the Section 16.6 salvage) but the time-average PPV need not equal the PPV at the time-average rates, because PPV is nonlinear in its arguments. The *Chernoff bound of chapter 11* (Theorem 11.4.1) assumed iid Bernoulli summands; with varying $\varepsilon(t)$ the X_t are independent but not identically distributed, and the multiplicative form that relied on a single p no longer applies directly. Its martingale generalization does.

Remark (Independent random rates leave composition unbiased)**.** *Random stage rates do not bias the composition formula when the stages are independent. If the η_i are independent with means $\mu_i = \mathbb{E}[\eta_i]$, the expectation of a product of independent factors is the product of the expectations, $\mathbb{E}[\prod_i(1-\eta_i)] = \prod_i(1-\mu_i)$, so $\mathbb{E}[1-\prod_i(1-\eta_i)] = 1-\prod_i(1-\mu_i)$: plugging the mean rates into the deterministic formula returns the exact expected combined error. What random rates add is variance, not bias. The error along the worst trajectory can still exceed the mean, as the section opened by noting, but in expectation the formula is exact. A directional bias appears only when the stages are correlated, the case Section 16.3 treats, where the sign of the bias follows the sign of the correlation.* △

The right tool: martingale concentration

The Azuma-Hoeffding inequality replaces the Chernoff bound off the iid path. Where Chernoff bounded sums of iid Bernoulli variables, Azuma-Hoeffding bounds martingale differences that are bounded in size but otherwise unrestricted in distribution.

Proposition 16.2.1 (Azuma-Hoeffding)**.** *Let $\{Y_t\}_{t=0,1,\dots,n}$ be a martingale with $Y_0 = \mathbb{E}[Y_n]$ and bounded differences $|Y_t - Y_{t-1}| \le c_t$ for each $t \ge 1$. Then for any $a > 0$,*

$$\Pr[|Y_n - Y_0| \ge a] \;\le\; 2\exp\!\left(-\frac{a^2}{2\sum_{t=1}^{n} c_t^2}\right).$$

The inequality is Hoeffding's (1963), with the martingale form from Azuma (1967); the MGF-plus-Markov argument of chapter 11 carries over once the per-step MGFs are bounded, and the proof never needs identical distributions, only the per-step jump bound. Applied to time-varying false-positive rates, let $X_t \in \{0,1\}$ indicate a false positive at query t with $\Pr[X_t = 1 \mid \mathcal{F}_{t-1}] = \varepsilon(t)$, and set $Y_t = \sum_{s\le t}(X_s - \varepsilon(s))$, a martingale with $|Y_t - Y_{t-1}| \le 1$. With $c_t = 1$,

$$\Pr\Big[\,\Big|\sum_{t=1}^{n} X_t - \sum_{t=1}^{n}\varepsilon(t)\Big| \ge a\,\Big] \;\le\; 2\exp\!\left(-\frac{a^2}{2n}\right),$$

the non-iid generalization of the Chernoff bound: the total false-positive count concentrates around $\sum_t \varepsilon(t)$ with Gaussian-shaped tails of width $O(\sqrt{n})$ even when $\varepsilon(t)$ is not constant.

Worked example: Bloom filter past capacity

Take the over-filled filter above, queried $n = 1000$ times after the fill with non-member inputs, the per-query rate having grown from 0.01 to 0.5 along the way. Treating the rate as a constant $\bar{\varepsilon} \approx 0.25$ and applying the iid Chernoff bound to the expected count 250 gives $\Pr[X \geq 275] \leq e^{-0.83} \approx 0.44$, a useless bound, because the constant-rate premise is wrong: its midpoint mean $0.25 \cdot 1000 = 250$ overstates the honest trajectory sum $\sum_t \varepsilon(t) \approx 230$ (the convex Bloom-fill curve spends most of its queries at low rates), so the bound is centered on the wrong count to begin with. The Azuma-Hoeffding bound assumes no constant rate: with $c_t = 1$, $\Pr[|\sum_t X_t - \sum_t \varepsilon(t)| \geq a] \leq 2e^{-a^2/2000}$, and $a = 100$ gives $2e^{-5} \approx 0.013$. The count lies within ± 100 of the honest trajectory expectation $\sum_t \varepsilon(t) \approx 230$ with probability at least 0.987. The iid analysis fails not because its bound is weak but because its model (constant rate) does not match the channel; the martingale analysis succeeds by acknowledging the non-stationarity.

Diagnostic

To decide between the two, sort the query log by time and test whether $\hat{\varepsilon}$ is stable across time-ordered partitions: split into early, middle, and late buckets and compare the three estimates. If they lie within their sampling intervals, the iid assumption is consistent and Chernoff applies; if they differ by more, stationarity is broken and Azuma-Hoeffding along the realized trajectory is the right tool. The test is cheap and catches the Axiom 2 failure specifically; it says nothing about the correlated-noise mode of Section 16.3 or the adversarial mode of Section 16.4, each of which needs its own check.

16.3 Correlated Noise Across Queries

The second failure mode trades a different axiom. Where Section 16.2 watched the per-query rate drift, this section holds the rate fixed and watches the independence between error events collapse. The channel matrix is stable in the sense of Section 4.2; what fails is the assumption that the error events are independent, and the joint distribution of outcomes no longer factors as the product of marginals.

The phenomenon

Three examples make the pattern concrete. The first is degenerate but instructive: one Bloom filter and one query, repeated. The response to $x \notin A$ is fixed by the bit pattern at k positions; once that pattern is set, the answer is deterministic, and the second query for the same x returns the same answer with probability one. The conditional probability of a second $\top$ given a first is 1, not the marginal ε, and the gap is the entire failure.

The second is the genuine Bloom correlation, internal to a single query. A query for x reads k bits $h_1(x), \ldots, h_k(x)$ and returns $\top$ only if all are set. These k bit-check events are not independent: a query's hashes can collide, and a queried bit being set is evidence that the array is fuller, which raises the chance the others are set too. The conjunction is correlated within the query.

The third introduces time. An online classifier updates its model after each response, so the classification at query t depends on the responses to queries $1, \ldots, t-1$ through the updated parameters. Sequential queries carry a Markov dependence on the model state; the per-query rate may be stable on average while the conditional distribution of the t-th error given the history is nontrivial.

Which axiom is violated

Section 4.2 stated Axiom 1 as the mutual conditional independence of error events at distinct inputs given the true classifications: for inputs $x_1, \ldots, x_n$ with true labels y_i and observed labels $\hat{y}_i$,

$$\Pr[\hat{y}_1 = a_1, \ldots, \hat{y}_n = a_n \mid y_1, \ldots, y_n] \;=\; \prod_{i=1}^{n} \Pr[\hat{y}_i = a_i \mid y_i].$$

This factorization underlies every multi-query bound in the book; when it fails the joint becomes an inequality, in either direction depending on the sign of the correlation. The three examples fail it in different ways: the repeated query fails completely (the conditional is degenerate, 0 or 1); the within-query bit-checks fail positively (one bit set raises the posterior that the others are); the online learner fails through a temporal Markov dependence. Axiom 1 can fail while Axiom 2 (Axiom 2's stability) holds; the two modes are independent.

Consequences for the iid tools

The composition theorem of Section 3.3 ($\eta_{\text{total}} = 1 - \prod_i (1 - \eta_i)$) used Axiom 1 the moment it wrote the product: the product of marginal no-error probabilities equals the joint no-error probability only under independence. Under positive correlation the joint no-error probability is larger (the system does better than the product predicts), under negative correlation smaller, so the formula becomes an approximation whose error direction tracks the correlation sign.

The Bloom derivation of Section 5.2 contains exactly this approximation. The standard $\varepsilon \approx (1 - e^{-kn/m})^k$ formula (Equation (5.1)) treats the k bit-check events of one query as independent, writing "all k set" as a product of per-bit probabilities. Line 9 of chapter 5 recorded that the factorization is an approximation: the k events are positively dependent, since a queried bit being set is evidence the array is fuller, so the conjunction is *larger* than the product and the standard formula *under*-estimates the false-positive rate. The gap vanishes as $m \to \infty$ with n/m fixed but is detectable at small m: the worked $m = 4$ witness of Line 9 gives an exact $13/64 \approx 0.203$ against the formula's 0.155. This is the Axiom 1 failure scoped within a single query; because the combination is a conjunction, the positive sign makes the product a *lower* bound, the opposite direction from the $1 - \prod_i (1 - \eta_i)$ serial form above, where positive correlation makes the product an upper bound on the joint error. The sign of the correlation and the form of the combination together fix the direction.

The PPV machinery of chapter 7 is a per-query quantity and is unaffected at a single query; the issue arises only when combining PPV estimates across queries, where the variance of the empirical PPV depends on the joint distribution of the contingency counts and no longer satisfies the iid sampling that Proposition 7.4.1 assumed.

The right tools

Three replacements are standard. *Conditional bounds via the law of total probability*: when the dependence has known structure, condition on the relevant variable and bound each conditional separately. For a query whose k bit-checks are coupled through the array occupancy, conditioning on the number of set bits makes the checks exchangeable and the conjunction computable. *McDiarmid's inequality* for bounded-difference functions:

Proposition 16.3.1 (McDiarmid's bounded-differences inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables and f a real-valued function satisfying the*

bounded-differences property: for each i there is a constant c_i with

$$\sup_{x_1,\ldots,x_n,x_i'} |f(x_1,\ldots,x_i,\ldots,x_n) - f(x_1,\ldots,x_i',\ldots,x_n)| \;\le\; c_i.$$

Then for any $a > 0$,

$$\Pr[|f(X_1,\ldots,X_n) - \mathbb{E}[f]| \ge a] \;\le\; 2\exp\!\left(-\frac{2a^2}{\sum_{i=1}^n c_i^2}\right).$$

The inequality is McDiarmid's (1989). Here f might be the empirical false-positive count of a fixed filter under a fixed query stream, a function of the hash-family choices; the function value moves by a bounded amount when one hash is resampled even though the bits a single hash sets are not independent, so the bounded-differences form gives a usable bound where the strict iid Chernoff bound does not apply. *Negative correlation as a salvage*: when the dependence is known to be negative, the iid bounds remain valid as upper bounds, since a negatively correlated system errs less than the iid analysis predicts. Sampling without replacement against sampling with replacement is the classic instance; identifying negative correlation lets one keep the iid analysis as conservative.

Worked example: when shared hashes do and do not correlate

Take two Bloom filters B_A and B_B built with a shared hash family $h_1,\ldots,h_k$, $k = 3$, $m = 100$ bits each, holding $n_A = n_B = 20$ items. The per-filter rate from Equation (5.1) is

$$\varepsilon \;=\; (1 - e^{-kn/m})^k \;=\; (1 - e^{-0.6})^3 \;\approx\; 0.092,$$

so the iid prediction for "both say $\top$" on a query $x \notin A \cup B$ is $\varepsilon^2 \approx 0.0085$. A natural worry is that the shared hash family correlates the two answers. It does not, when A and B are independent. For a fixed query x both filters read the same positions $h_1(x), h_2(x), h_3(x)$, but B_A's bits at those positions depend only on A and B_B's only on B; with A and B independent the two bit patterns are independent, so the answers are independent and the joint is the product exactly. Computing position by position confirms it: a given position is set in B_A with probability $p_A \approx 0.45$ and independently in B_B with $p_B \approx 0.45$, set in both with $p_A p_B \approx 0.20$, and all three set in both with $(p_A p_B)^3 \approx 0.0085$, the same ε^2. Sharing the hash family fixes *which* positions are read, not *whether* the two independent arrays have them set.

Correlation appears when the sets overlap. If A and B share elements, those elements set the same bits in both filters, coupling the answers; in

the extreme $A = B$ the two filters are identical and the joint probability of two $\top$ responses is the marginal ε, not ε^2, the iid prediction undershooting by a factor $1/\varepsilon \approx 11$. The lesson sharpens the section: shared randomness is necessary but not sufficient for correlated error. The error events must *jointly* depend on it, as the k bit-checks within one query do (through the shared array occupancy) and as two filters on overlapping sets do (through the shared elements), but as two filters on independent sets do not.

Remark (The composition formula brackets the truth)**.** *The sign analysis gives the designer who knows only the marginal stage rates a two-sided guarantee. Under* negative *association of the stage errors (*$\Pr[\textit{error at } i \textit{ and } j] \leq \Pr[\textit{error at } i]\Pr[\textit{error at } j]$ *for every pair), the serial composition formula* $1 - \prod_i (1 - \eta_i)$ *is an* upper *bound on the true combined error rate; under* positive *correlation it is a* lower *bound, as the within-query Bloom conjunction above shows. The iid formula therefore sits between the two correlated regimes and brackets the true combined rate: a designer with only marginals knows the truth lies on the conservative side of the formula once the correlation sign is known, and within a computable interval when it is not.* $\triangle$

The diagnostic for correlated noise is to ask whether the error events share a source they *jointly* depend on: the same array occupancy, the same elements, the same random seed consumed at query time. If they do, the iid analysis is an approximation whose direction and size need explicit calculation; Section 16.6 returns to which parts of the framework survive the correlation.

16.4 Adversarial Inputs

The third failure mode breaks both axioms at once. Where Section 16.2 watched the channel drift through nature and Section 16.3 watched queries carry shared noise, this section watches queries chosen with intent. An adversary observes the responses and selects subsequent queries to expose the worst case, and the probabilistic setup, which presumed a query distribution the analyst could pin down in advance, gives way to a worst-case strategy that depends on the adversary's goals and knowledge.

The phenomenon

A query stream is adversarial when the t-th query depends on the responses to queries $1, \ldots, t-1$ and is chosen to maximize an attacker objective: the empirical false-positive rate, a side-channel signal, a successful collision. The

gap from the iid model can be stark. A Bloom filter sized for $\varepsilon = 0.01$ gives an iid attacker a 1% chance of a false positive per random query; an attacker who learns the hash family can compute $h_1(x), \ldots, h_k(x)$ for candidates and select inputs whose images land at already-set positions, pushing the per-crafted-query rate close to 1. The Bernoulli analysis described a rate of 0.01; the adversarial reality is a rate near 1 per crafted query. The gradient-descent attacker against a neural classifier is the same phenomenon at scale: the empirical attack rate against current image classifiers is essentially 100% with small perturbations, while the benchmark ε described natural inputs, not adversarial ones.

What is violated

Both axioms fail, and they stack. Axiom 1 (Axiom 1) presumes the inputs are conditionally independent given the true labels; the attacker's t-th query is by construction a deterministic function of the prior responses, so the inputs are not independent. Axiom 2 (Axiom 2) fails more subtly: the per-block rates are defined relative to some input distribution, and an adversary selecting inputs at the tail of that distribution replaces the expected-case per-block rate with the worst-case one. The cleanest summary is that the worst-case attacker converts the analyst's expectation-based statements into supremum-based ones: where the analyst wrote $\mathbb{E}_x[\varepsilon(x)]$ for an average over a presumed distribution, the attacker reads $\sup_x \varepsilon(x)$ as the rate that obtains, and the two can differ by orders of magnitude.

Concrete examples

Bloom filter evasion. An attacker who knows the hash family can compute, for any candidate y, the positions B_A would read, and if those are all set (because other items in A set them), the filter returns $\top$ on $y \notin A$. Random sampling from the universe finds such a y in $1/\varepsilon$ tries; a structured search targeting positions known to be set finds one in a constant number of attempts.

Side-channel timing. A filter that returns $\perp$ as soon as one of the k bits is found unset answers faster on a clear bit; an unusually slow response signals that all k bits were checked, that is, a $\top$ and a candidate false-positive target. An attacker who measures response time localizes the attack region in a far smaller candidate space. The framework's analysis described the bit-pattern probabilities, not the access pattern; timing sits outside its scope.

Adversarial machine learning. A classifier at a benchmark ε on held-out data faces an attacker who follows the confidence gradient to inputs near decision boundaries. The empirical worst-case error under gradient attacks is essentially 100% for many natural classifiers, regardless of the benchmark value; the benchmark and the worst-case statements describe different input distributions, and conflating them is the failure.

Cryptographic chosen-plaintext attacks. In the encrypted-search context of Chapter 18, an attacker who chooses plaintexts and observes index responses is far stronger than one restricted to the natural document distribution. The Bernoulli analysis assumed that distribution; chosen-plaintext attacks let the attacker pick the worst-case document, the same expectation-versus-supremum gap. Chapter 18 catalogues the cryptographic defenses (semantic security, indistinguishability under chosen-plaintext attack); this section flags the structural form of the attack rather than the remediation.

The right tools

Off the iid path the analyst's replacements are worst-case bounds and cryptographic primitives. *Worst-case bounds* drop the expectation and bound the maximum: a Bloom filter's worst-case ε over inputs is set by its bit pattern, a classifier's worst-case loss over a perturbation ball by the model's Lipschitz constant times the radius. These are loose against natural-distribution expectations but are the right object, describing what the attacker controls. *Cryptographic hash families* make the family secret: the Bloom hashes are generated from a secret salt, the filter publishes only the bit pattern, and an attacker who wants to construct false positives must first guess the salt, which under standard assumptions costs $2^{|\mathsf{salt}|}$ work. The structural attack collapses to a brute-force key search, and the false-positive rate under salt-secret operation matches the iid analysis provided the salt has enough entropy. This is the cryptographic core of cipher maps; Chapter 17 takes up the details. *Trapdoor primitives* handle the side channels: when the attacker can observe timing, memory access, or control flow, the right machinery is trapdoor computing, whose data-oblivious algorithms have access patterns that do not depend on the data they process, with oblivious RAM and constant-time cryptographic operations the classical instances. Chapter 17 details these as the bridge between Bernoulli analysis and adversarial environments; the framework continues to apply if and only if the side channels are closed.

Worked example: Bloom filter evasion

Take a filter sized for $\varepsilon = 0.01$ at $k = 7$, $m/n = 9.585$, $n = 1000$ items. The bit pattern has a fraction $1 - e^{-kn/m} = 1 - e^{-0.73} \approx 0.518$ of bits set at design fill, so a random non-member is a false positive with probability $0.518^7 \approx 0.010$, matching the design target. The *iid attacker*, querying uniformly at random, waits $1/\varepsilon = 100$ queries on average for a false positive: a diffuse, undirected adversary. The *structural attacker*, who knows the hash family and reads the published bit pattern, examines candidates offline and checks locally whether all k positions are set; a random candidate works with probability $0.518^7 \approx 0.010$, so the attacker still examines about 100 candidates, gaining only the ability to do so offline rather than against the live filter. Disclosure of the family alone buys almost nothing here. The *targeted attacker* does the real damage: by solving a small system of constraints to construct an input y whose hashes land at chosen set positions $j_1, \ldots, j_k$, the attacker manufactures false positives in $O(1)$ work per attempt, and the practical attack proceeds at machine speed.

The gap from iid analysis to worst-case adversarial analysis thus runs from essentially nothing (the structural attacker exploiting only disclosure) to several orders of magnitude (the targeted attacker constructing inputs). In any security model that grants the attacker knowledge of the hash family, the iid ε is not a meaningful security parameter; the salt-secret construction or a trapdoor primitive must be invoked. Section 16.6 returns to which framework constructs keep their traction once those defenses are in place.

16.5 When No Latent Value Exists: Quantum Contextuality

The failure modes catalogued so far share a feature that is easy to overlook: in every one, a definite latent value still exists. A drifting rate (Section 16.2) is the rate of a bit that is genuinely 0 or 1; correlated noise (Section 16.3) garbles genuine answers with a dependence the factorization missed; an adversarial input (Section 16.4) has a true membership status, chosen to sit at the tail. In each case the framework mis-estimates a parameter of a process whose underlying value is never in question. There is one failure of a different kind, where the assumption beneath both axioms fails: there is no consistent latent value to estimate at all. Quantum mechanics is its rigorous physical instance, and it gives this chapter its sharpest example.

The latent value is a hidden variable

The framework's atom posits a definite truth t, garbled by a channel Q and recovered by Bayes. Read literally, t is a *non-contextual local hidden variable*: the answer to "is x a member?" is a pre-existing fact, and Q only obscures it. This commitment is stronger than either axiom and prior to both. It asserts that a single joint distribution over the answers to all queries exists, of which each observed rate is a marginal. Axiom 1 (Axiom 1) then adds that this joint factorizes across instances; Axiom 2 (Axiom 2) adds that the per-block rates are stable across queries. The existence of the joint distribution is assumed before either axiom is written, and it is the assumption this section examines.

Fine's theorem: the latent value is a falsifiable claim

That the latent value exists looks like a harmless modeling convenience. It is not. Fine [Fin82] proved that, for a correlation experiment, the following are equivalent: a deterministic hidden-variable model exists; a factorizable stochastic model exists; a single joint distribution over all of the experiment's observables exists and returns the measured probabilities; and the Bell inequalities hold. Translated into the framework's vocabulary, the chain reads

a latent value exists $\iff$ a joint distribution over all answers exists $\iff$ the Bell inequalities hold.

The latent-value assumption is therefore a falsifiable physical claim, not a definitional choice, and it can be tested in the laboratory.

The physical archetype

Quantum mechanics falsifies it. Two entangled qubits, measured in suitably chosen bases, produce correlations that violate the Bell inequalities [Bel64]. By Fine's equivalence no joint distribution over the measurement outcomes exists, so the pair cannot be written as two Bernoulli[bool] values, not with any rates and not with any correlation correction. This is the precise difference from Section 16.3: correlated noise widened the composition theorem into an inequality with correction terms (Section 16.3) because a joint distribution still existed and merely failed to factorize; entanglement removes the joint distribution itself, so there is nothing left for a correction term to correct.

A second, independent obstruction appears once measurements can be incompatible and the state space has dimension at least three. The Kochen-Specker theorem [KS67] shows that no assignment of definite values to all observables can be made consistent across the measurement contexts in which those observables appear. The honest scope matters here: a single two-outcome measurement in a fixed basis does admit a hidden-variable model, so the atom in isolation is safe; the obstructions are entanglement (Bell) and dimension at least three (Kochen-Specker).

Reading the axioms as physics

Seen through the framework, Axiom 1 is a locality assumption, and the Bell inequality is the exact quantitative bound on how far an entangled pair sits outside any factorized Bernoulli composition. The pre-axiom existence of a joint distribution is non-contextuality. The negative space of this chapter has, in quantum contextuality, its most rigorous possible example. The earlier failures are cases where the framework estimates a real parameter badly; this is a setting where the framework's ontology is provably wrong, with a theorem certifying the impossibility rather than an experiment merely exhibiting a hard case.

Where the boundary sits

For the earlier failures the chapter could name a replacement: a martingale bound, a correction term, a secret hash family. Here there is none within the classical framework. The correct object is a probability amplitude, not a probability, and amplitudes can cancel where probabilities can only accumulate; the interference this permits has no representation in a latent-value-plus-noise model. In the density-matrix description [NC10] the framework sees only the diagonal of the state in the measurement basis, and the off-diagonal coherences that carry interference are invisible to it.

The boundary is therefore sharp and locatable rather than a vague caution. The framework is exact precisely where the state is diagonal in the measurement basis: an eigenstate, a classical mixture, or a decohered state. That is the regime of classical readout, and decoherence is the physical process that zeroes the off-diagonals and hands the framework a genuine latent value to model. The forward error calculus of that readout regime, in which measured qubits are reported through an asymmetric (ε, ω) channel and Boolean post-processing composes the rates, is the subject of the companion paper [Tow26a]. In this precise sense the Bernoulli model is the classical

shadow of a quantum system: faithful on the diagonal, blind to the rest. Section 16.6 takes up what survives at this boundary and elsewhere.

16.6 Where the Framework Still Helps

The three preceding sections enumerated systems for which strict iid analysis breaks. The pessimistic reading, that the framework is useless once either axiom fails, is too strong. The framework's geometric and structural content is per-query, and the per-query content survives the failures of Axiom 1 or Axiom 2 with only minor accommodation; what disappears is the right to chain per-query rates into multi-query bounds without further work. This section catalogues the partial successes.

The perfect-filter line classifies per-query

Definition 6.4.1 read the perfect-filter property, $\varepsilon = 0$ or $\omega = 0$ exactly, off a single channel matrix. The classification was set up for stable rates, but it survives drift and correlation because it acts on per-query marginals. A system whose rates drift still has a $(\varepsilon(t), \omega(t))$ point at each query and traces a trajectory in $[0,1]^2$, and the line classifies each point: a Bloom filter past capacity has nonzero false-positive rate at every time but stays on the $\omega = 0$ axis throughout, never producing a false negative. Correlation between queries shifts the joint away from the iid product but moves neither marginal off its axis. A designer can use the classification to reason about whether a noisy bit ever errs toward $\top$, ever toward $\bot$, or both, even under drift or correlation; only the chaining of per-query rates into multi-query bounds needs further work.

The channel matrix becomes a sequence (Q_t)

Definition 9.2.1 fixed the channel matrix Q as the per-block conditional probabilities of observed given true labels. Under Axiom 2's failure Q becomes a sequence (Q_t), one matrix per query, and the framework absorbs the generalization at the per-query level: the eigenstructure, the channel capacity, and the Bayes-optimal decision rule at query t all read off Q_t alone. The Kronecker theorem (Theorem 9.3.1) generalizes the same way, applied sliced or conditioned rather than unconditionally: for time-varying rates it gives a sequence of Kronecker products, one per step; for correlated channels the factorization holds conditional on the shared randomness that drives the

correlation. The geometry is portable; the stability that let a sequence of queries be treated as draws from one matrix is what is lost.

The PPV/NPV formula stands per-query

Definition 7.4.1 expressed PPV from ε, ω, and prevalence π at a single query, and nothing in the derivation used iid sampling across queries. Under any of the three failure modes the formula still holds at each query; what changes is that the triple may itself be query-dependent. Time-averaging needs care, because PPV is nonlinear in π, so the time-average PPV is not the PPV at the time-average rates, but at any single query the formula returns the correct conditional probability. The practical implication is to report PPV and NPV per-query or per-window rather than as a single global summary; a designer who sees π drift can plot PPV over time and reason about its trajectory without committing to a fixed π the data would contradict.

A pragmatic rule

The catalogue gives a working rule in two halves. First, keep the geometry and the channel-matrix view even when both axioms fail: the perfect-filter line classifies per-query, the channel matrix factorizes per-query or per-block of independent channels, and the PPV/NPV formulas evaluate per-query, none of which assumes iid. Second, replace the iid tools with their non-iid counterparts: where Theorem 11.4.1 bounded sums of iid Bernoulli variables, use Proposition 16.2.1 for the martingale setting of Section 16.2 or Proposition 16.3.1 for the bounded-differences setting of Section 16.3; where neither iid nor a tractable dependence applies (Section 16.4), drop the expectation for worst-case bounds and close the side channels with cryptographic or trapdoor primitives. The framework's affirmative content is a collection of per-query objects, a noisy bit with rates (ε, ω), a channel matrix Q, a derived quantity such as PPV. Strict iid glues them into multi-query bounds; when iid fails the per-query objects survive and only the glue needs replacing.

16.7 What's Next

This chapter mapped the negative space around the two axioms in three modes, each named by the mechanism that pushes a system off the iid regime: varying rates over time (Section 16.2), correlated noise across queries (Section 16.3), and adversarial inputs (Section 16.4). Section 16.6 recovered the per-query content that survives each failure, leaving a working rule

rather than a binary verdict: the geometry and channel-matrix view apply per-query, and the multi-query bounds give way to their non-iid counterparts.

The chapter's placement is structural. It opens Part V between the constructive culmination of Part IV, the BHF and BPHF inside the framework's domain of validity, and the cryptographic applications of chapters 17 and 18, which extend the framework into settings where clean iid analysis can no longer be relied on. Without the boundary chapter, those applications would inherit the iid assumptions implicitly and the guarantees they advertise would lean on hidden conditions.

The two chapters that close Part V are direct heirs to the boundary work. Chapter 17 develops cipher maps and trapdoor computing, recasting the BHF and BPHF architecture for settings where the perfect hash is replaced by a trapdoor function whose forward direction is public and reverse direction private. Chapter 18 specializes the machinery to encrypted search, where the FPR and FNR guarantees of Part IV carry over and the cryptographic context demands additional structure (data-oblivious access patterns, side-channel control). Both live in territory this chapter flagged as non-trivial: encrypted-search queries are adversarial by construction, the server curious and the queries chosen by an attacker with access to the index, so the worst-case tools of Section 16.4 replace iid expectations and the secret hash family closes the side channels the section named. The analytical alternatives the chapter invoked, the martingale and bounded-differences inequalities and the worst-case and cryptographic tools, are anchored in Section 16.7.

Bibliographic Notes

Martingale concentration. The Azuma-Hoeffding inequality of Proposition 16.2.1 has two parents: Hoeffding [Hoe63], "Probability Inequalities for Sums of Bounded Random Variables," gave the independent-case predecessor, and Azuma [Azu67], "Weighted Sums of Certain Dependent Random Variables," the bounded-difference martingale generalization, the two sharing the exponential-moment proof technique that fuses them into one inequality. The extension to non-martingale functions of dependent variables is McDiarmid [McD89], "On the Method of Bounded Differences," the reference for Proposition 16.3.1 and the clearest expository treatment of the technique.

Non-iid concentration in learning theory. The textbook reference for non-iid concentration as used in statistical learning theory is Devroye and Lugosi [DL01], *Combinatorial Methods in Density Estimation*, which catalogues the martingale, bounded-difference, and chaining techniques applied to empirical processes. Its worked examples (kernel density estimators, histogram classifiers, nearest-neighbor rules) transcribe into the Bernoulli setting with minor modification.

Adversarial machine learning. The adversarial-inputs catalogue of Section 16.4 draws on the literature that grew around neural classifiers since the mid-2010s. The two entry points are Goodfellow, Shlens, and Szegedy [GSS15], "Explaining and Harnessing Adversarial Examples," which gave the fast-gradient-sign method and named the phenomenon, and Carlini and Wagner [CW17], "Towards Evaluating the Robustness of Neural Networks," whose optimization-based attack became the standard benchmark for defenses. The Bernoulli framework inherits the same exposure: an attacker who knows the hash family finds a colliding input in expected $O(1/\varepsilon)$ work, and the honest defenses are secret hash families or trapdoor primitives.

The cryptographic foundations. The primitives the chapter points to as replacements for iid analysis in the adversarial regime are catalogued in the trapdoor-computing and secure-computation literatures, the foundational treatment being Goldreich [Gol96], "Foundations of Cryptography." The 1996 fragments (and the 2001 and 2004 volumes that grew from them) are the entry point for secure multiparty computation, oblivious RAM, and the private-information-retrieval primitives chapter 17 unpacks into the cipher-map framework. Readers anticipating chapters 17 and 18 should treat Goldreich's foundations as the prerequisite; the Bernoulli-framework specialization sits on top of the cryptographic infrastructure those volumes formalize.

Quantum contextuality. The boundary of Section 16.5, the regime where no latent value exists, rests on three results. Bell [Bel64], "On the Einstein Podolsky Rosen Paradox," gave the inequality whose violation rules out local hidden variables; Fine [Fin82], "Hidden Variables, Joint Probability, and the Bell Inequalities," proved the equivalence the section turns on, that a joint distribution over all observables exists exactly when the Bell inequalities hold; and Kochen and Specker [KS67], "The Problem of Hidden Variables in Quantum Mechanics," established the contextuality obstruction in

dimension at least three. The density-matrix and decoherence facts that place the framework on the diagonal (readout) sector are standard; Nielsen and Chuang [NC10], *Quantum Computation and Quantum Information*, is the textbook reference.

Chapter 17

Cipher Maps and Trapdoor Computing

17.1 What is a Cipher Map?

The first sixteen chapters analyzed noisy computation on cleartext: a Bloom filter errs because the structure is small, a classifier because the world is hard, a sketch because the stream is large, and in each case the user accepts the error to save space or time. This chapter applies the same machinery to a different setting. The data is encrypted, the computation runs on the encrypted form, and only a holder of a secret key can read the result. Bloom filters and noisy classifiers reappear, but now their false positives and false negatives carry a second role alongside the operational one: the bounded error is also what keeps the plaintext hidden from the party doing the computation.

The abstraction

Begin with a plaintext function $f : X \to Y$. In the cleartext setting a user holds x, applies f, and reads $f(x)$. The cipher-map setting introduces a split. A holder of x (the *client*) wants $f(x)$ but will not give x to the machine computing it. The client encrypts x under Enc, sends the ciphertext to a *server*, and asks the server to compute on $\mathsf{Enc}(x)$ something the client can decrypt to (something close to) $f(x)$. The server never sees x, and never sees $f(x)$ directly; what it returns is an encryption of an output the client decrypts with Dec. The companion map that operates on the encrypted representations is the cipher map.

Definition 17.1.1 (Cipher map)**.** *Let $f : X \to Y$ be a plaintext function, $\mathsf{Enc} : X \to \mathsf{Enc}(X)$ an encryption map with decryption $\mathsf{Dec} : \mathsf{Enc}(Y) \to Y$. A* cipher map *for f is a function $\tilde{f} : \mathsf{Enc}(X) \to \mathsf{Enc}(Y)$ on the encrypted representations such that, for $x \in X$,*

$$\Pr\Big[\,\mathsf{Dec}(\tilde{f}(\mathsf{Enc}(x))) = f(x)\,\Big] \;\geq\; 1 - \omega$$

when $f(x)$ is a positive output, and the probability of a spurious $y \neq f(x)$ on the negative case is at most ε. The pair (ε, ω) is the cipher map's error profile*, in the Bernoulli sense of Section 1.2.*

The cipher map operates entirely on the encrypted domain; decryption happens client-side after the encrypted output returns; and the rates ε, ω are exactly those the framework has spent sixteen chapters analyzing, so there is no new probabilistic machinery. What is new is the second meaning the error carries. A cipher map with $\varepsilon = \omega = 0$ behaves identically to f from the client's view but also reveals the most about x to an adversary watching $\tilde{f}$ across many queries; a cipher map with positive error obscures x in proportion to the error. Section 17.2 makes the duality precise and Section 17.3 unpacks its consequences.

Why approximate is acceptable

The exact alternative exists. Fully homomorphic encryption (FHE), introduced by Gentry in 2009 and refined by the BFV, BGV, and CKKS schemes, gives an $\tilde{f}$ with $\varepsilon = \omega = 0$ for any computable f. The catch is cost: FHE ciphertexts run kilobytes to megabytes per encrypted bit and seconds per bit operation, six orders of magnitude slower and three larger than the plaintext computation for microsecond workloads. Cipher maps trade error for practicality, just as the cleartext approximate-set literature trades Bloom false positives for memory. A Bloom-based cipher map for membership uses about $-\log_2 \varepsilon + \mu$ bits per element (the optimal formula of chapter 14), runs in constant time, and gives a tunable false-positive rate. Section 17.4 works the comparison in detail.

The trapdoor key

What distinguishes a cipher map from any other noisy function is the *trapdoor key*: a secret that recovers $f(x)$ from $\tilde{f}(\mathsf{Enc}(x))$. The client holds it; the server does not. Without the trapdoor the server sees only opaque bit patterns and cannot tell meaningful queries from filler, real outputs from noise,

or one plaintext from another; with it the client decrypts to within the (ε, ω) bounds. Three properties make the key worth its name: it is *short* (constant length, not growing with the data), *necessary* (no polynomial-time computation without it recovers plaintext), and *sufficient* (a key holder decrypts at the same per-call cost as a plaintext evaluation). The three distinguish a cipher map from a generic encryption (which lacks sufficiency) and from a generic noisy function (which lacks the first two).

Three example cipher maps

Three cipher maps recur in this chapter and chapter 18, sketched here to ground Section 17.2.

Encrypted set membership. For a plaintext set $A \subset X$ the client wants $f_A(x) = \mathbf{1}[x \in A]$ without revealing A. It builds a Bloom filter on A with a keyed hash family (the key is the trapdoor), encrypts the bit array, and gives it to the server; to query, it sends $\mathsf{Enc}(x)$ and the server tests the encrypted bits. The returned bit is correct on negatives with probability at least $1 - \varepsilon$ and always correct on positives ($\omega = 0$), the one-sided error of the Bloom construction.

Encrypted equality test. For $f_=(x, y) = \mathbf{1}[x = y]$, deterministic encryption makes identical plaintexts encrypt identically, so ciphertext equality coincides with plaintext equality and $\varepsilon = \omega = 0$. The cost is a known leak: an adversary watching many tests builds the plaintext distribution from ciphertext frequencies. This is a cipher map with no Bernoulli error but nonzero information leak, a reminder that the rates are not the only confidentiality knob.

Encrypted ordering. For $f_\leq(x, y) = \mathbf{1}[x \leq y]$, order-preserving encryption preserves plaintext order in the ciphertext encoding, so the server compares ciphertexts directly at $\varepsilon = \omega = 0$, leaking the entire plaintext ranking, a larger leak than equality.

The three span the landscape. The Bloom-based map sits at the high-ε, zero-ω corner of the perfect-filter line of Section 6.4; the equality and ordering maps sit at the zero-zero corner but pay their confidentiality cost through structural leakage rather than rate. The chapter's work is to show that the (ε, ω) machinery grips the first case and that the PPV/NPV machinery of Definition 7.4.1 gives a confidentiality measure covering the second.

17.2 The Bernoulli Framework as Error Model

Definition 17.1.1 stated the abstraction in terms of a false-positive rate ε and a false-negative rate ω. That notation was not cosmetic: the rates are the same objects chapters 1 and 4 introduced for the Bernoulli atom, and a chained cipher-map composition obeys the same laws as a chained noisy-classifier composition. This section makes the equivalence operational, so the algebraic identities the framework proved earlier carry over to cipher maps unchanged.

Cipher-map rates are Bernoulli rates

A Bernoulli predicate has per-query behavior summarized by $\varepsilon = \mathrm{P}\left(\tilde{A}(x) = 1 | x \notin A\right)$ and $\omega = \mathrm{P}\left(\tilde{A}(x) = 0 | x \in A\right)$, and Section 4.5 showed that the two rates with the input prevalence specify the joint distribution of any finite query sequence. A cipher map $\tilde{f}$ for a Boolean predicate f has the same per-query behavior. The client's only observable is the decrypted bit $\mathsf{Dec}(\tilde{f}(\mathsf{Enc}(x))) \in \{0, 1\}$, and that bit is Bernoulli conditional on the plaintext class:

$$\varepsilon = \mathrm{P}\left(\mathsf{Dec}(\tilde{f}(\mathsf{Enc}(x))) = 1 | f(x) = 0\right), \quad \omega = \mathrm{P}\left(\mathsf{Dec}(\tilde{f}(\mathsf{Enc}(x))) = 0 | f(x) = 1\right),$$

the probability taken over $\tilde{f}$'s internal randomness (hash choice, ciphertext noise, key schedule). Encryption and decryption drop out because $\mathsf{Dec} \circ \mathsf{Enc}$ is the identity on correctly decrypted outputs, leaving a Bernoulli atom with the cipher map's error profile as its (ε, ω). Every theorem of chapters 1 through 15 that took (ε, ω) as input applies when the rates come from a cipher map; the framework is indifferent to where the error originates and tracks only how it propagates. A trapdoor set is in this sense a Bernoulli$[T]$ over its plaintext type T (Section 13.2), the encryption layer changing the representation but not the Bernoulli-square point.

Per-query channel matrices

The two-rate summary is convenient but limited. A Boolean cipher map is fully captured by a 2×2 channel matrix

$$Q = \begin{pmatrix} 1 - \varepsilon & \varepsilon \\ \omega & 1 - \omega \end{pmatrix},$$

rows indexed by the true plaintext class, columns by the observed output; the Bloom-based map has $\omega = 0$, a zero in the lower-left. For non-

Boolean cipher maps (encrypted classifiers, ranked retrieval) the 2×2 matrix generalizes to the $K \times L$ channel matrix of Definition 9.2.1, with $Q_{k,\ell} = \mathrm{P}\left(\mathsf{Dec}(\tilde{f}(\mathsf{Enc}(x))) = \ell | f(x) = k\right)$. The channel-matrix representation captures the full per-query behavior in one object that composes by multiplication, and it makes contact with the information-theoretic machinery of chapter 8 that Section 17.6 needs to quantify confidentiality.

Chained composition is matrix multiplication

Suppose the client computes $h = f \circ g$ on encrypted data, $X \xrightarrow{g} Y \xrightarrow{f} Z$: a cipher map $\tilde{g}$ returns an intermediate $\mathsf{Enc}(Y)$, then $\tilde{f}$ computes on it and returns $\mathsf{Enc}(Z)$. The error compounds. If $\tilde{g}$ produces a wrong $y' \neq g(x)$, then $\tilde{f}$ is asked to compute $f(y')$, and even an errorless $\tilde{f}$ returns $f(y') \neq f(g(x))$: the stage-1 error propagates through stage 2. This is the situation the chapter 4 Kronecker theorem (Theorem 4.4.1) and its n-ary generalization (Theorem 9.3.1) were built to analyze, and the factorization is matrix multiplication.

Theorem 17.2.1 (Cipher-map composition)**.** *Let $\tilde{g}$ and $\tilde{f}$ be cipher maps realizing $g : X \to Y$ and $f : Y \to Z$ with per-query channel matrices $Q_g \in [0,1]^{|X|\times|Y|}$ and $Q_f \in [0,1]^{|Y|\times|Z|}$. If Axiom 1 (Axiom 1) holds, so the two stages' errors are mutually independent given the inputs, then the composed cipher map $\tilde{f} \circ \tilde{g}$ realizes $h = f \circ g$ with channel matrix*

$$Q_h \;=\; Q_g\, Q_f,$$

ordinary matrix multiplication, $Q_h \in [0,1]^{|X|\times|Z|}$.

The proof reuses the chapter 4 argument: $(Q_g Q_f)_{x,z}$ sums over intermediate y the probability that stage 1 produces y times the probability that stage 2 then produces z, the conditional independence being Axiom 1 applied to the two-stage sequence. For Boolean maps the theorem is a 2×2 product; the same result governs any finite chain by induction, the composed matrix being the product of the per-stage matrices. A consequence: chained cipher-map error grows predictably, the composed Boolean rates bounded by chapter 3's $1 - \prod_i (1 - \eta_i)$ on each error type. This is Property 4 of the cipher-map design in the trapdoor-computing literature, the framework's composition law and the cryptographic side's requirement being the same property.

The geometry carries over

The perfect-filter line of Definition 6.4.1 classifies approximate sets by their (ε, ω) position, and the same classification applies to cipher maps. A cipher map with $\omega = 0$ is *safe*: it never misses a real positive, the client losing no real results, at the cost of $\varepsilon > 0$. The Bloom-based map sits on this line, as does any cipher map built by encrypting a perfect-filter-line approximate set. A cipher map with $\varepsilon = 0$ sits on the opposite axis, *exact when positive* but missing some real positives; it appears in randomized-encoding maps that occasionally reject valid plaintexts. The classification is geometric and applies to cipher maps as a special case of approximate functions.

Worked example: two-stage encrypted classifier

A representative composition appears in encrypted retrieval. The client retrieves and ranks documents matching an encrypted query in two stages. *Stage 1, encrypted membership*: the server holds an encrypted Bloom filter on a candidate set A, returns an encrypted membership bit at $\varepsilon_1 = 0.01$, $\omega_1 = 0$, so

$$Q_g = \begin{pmatrix} 0.99 & 0.01 \\ 0 & 1 \end{pmatrix}.$$

Stage 2, exact ranking on positives: an FHE-based ranking introduces no error, $Q_f = I$. By Theorem 17.2.1, $Q_h = Q_g I = Q_g$: the composed map inherits $\varepsilon = 0.01$, $\omega = 0$. Had stage 2 carried its own FPR-only error $\varepsilon_2 = 0.005$, the product would give composed $\varepsilon = 0.99{\cdot}0.005{+}0.01{\cdot}1 \approx 0.015$, matching $1 - (1 - 0.01)(1 - 0.005) \approx 0.015$, with ω staying zero because both stages are one-sided. The framework's composition law applies to cipher maps with no modification. The next section unpacks the operational consequences: when ε, ω are also confidentiality parameters, the same matrix algebra governs how confidentiality accumulates across composed queries.

17.3 The Trapdoor Computing Paradigm

The cipher map of Definition 17.1.1 is the mathematical object; the trapdoor computing paradigm is the operational setting in which it earns its keep. The setting has three commitments: the data is encrypted before it leaves the user, the computation runs entirely on the encrypted form, and the result can be read only by a holder of a short secret key, the trapdoor. The cipher map carries out the computation and the trapdoor unlocks the

answer; everything else, the plaintext records, the queries that select among them, the intermediate results, stays hidden from any party without the key.

The paradigm

The paradigm predates the modern encryption-on-the-cloud framing. Goldreich's work on software protection in the late 1980s asked how a program could run on an untrusted machine without revealing what it computed; the same question, in other language, drives oblivious RAM, secure multiparty computation, and private information retrieval. What unites them is the conviction that *compute* and *know* can be separated: a machine can do useful work on data whose meaning it cannot see. Trapdoor computing names the separation. The system has two distinguished resources, an encrypted dataset and a trapdoor key, both needed to read anything meaningful: the dataset without the key is opaque, the key without the dataset useless. Computation is the operation of cipher maps on the encrypted dataset, producing encrypted outputs the key holder can decrypt and no one else can. The Bernoulli framework enters at decryption: because the cipher map is approximate, the decrypted output is a Bernoulli observation rather than a deterministic fact, and the chapter 4 axioms govern how it behaves across queries. The cryptographic side sets up the problem (compute without revealing); the Bernoulli side tracks the consequence (read with controlled error).

Two-party setup

The cleanest instance has a *server* holding the encrypted dataset and running the cipher maps, and a *client* holding the trapdoor and issuing queries. The server sees ciphertexts in and ciphertexts out, nothing else. The encryption keeps the query private; the cipher map's bounded error keeps the dataset partially hidden even from a client who probes many times. The standard threat model is the *honest-but-curious* server, which follows the protocol but records the encrypted traffic for later analysis; bounding what that analysis can recover is the Bernoulli framework's job. A malicious server that deviates from the protocol needs cryptographic integrity checks beyond this chapter. The client's recovery quality is set by the (ε, ω) profile: an $\omega = 0$ map loses no real positives but admits some false ones, an $\varepsilon = 0$ map trusts every positive but misses some real ones.

Confidentiality versus correctness

The trade-off that distinguishes the paradigm from cleartext computation is between confidentiality and correctness. Tight error rates make the client's view faithful but turn each near-deterministic answer into a side channel: a server watching many such queries ties encrypted queries to the records they match and reconstructs the plaintext distribution. Loose rates make each query noisier for the client but push the encrypted-output distribution toward uniform across the dataset, defeating frequency analysis. The (ε, ω) point is therefore a confidentiality knob as well as a correctness knob, and the perfect-filter line of Definition 6.4.1 reads naturally in this light: an $\omega = 0$ map concentrates its confidentiality cost in the single parameter ε (every real positive returned, recall preserved), an $\varepsilon = 0$ map in ω (every positive answer correct, but real positives missed). The first suits recall-sensitive retrieval, the second precision-sensitive contexts. Section 17.6 returns with a quantitative measure.

Multiple queries: leakage accumulates

A single query reveals little; a million can reveal much. Under Axiom 1 (Axiom 1) the error events of m independent queries are mutually independent, and the joint distribution factors as the Kronecker product of m copies of the per-query channel matrix (Theorem 4.4.1). The same algebra describes the trapdoor stream, and any single-query confidentiality bound lifts through tensor algebra to the stream. The independence cuts both ways: the client's recovery sharpens as $m^{1/2}$ by standard concentration, and the server's information about the plaintext grows linearly in m for any map with positive single-query leakage. The two scalings are the same data-processing inequality read from opposite sides, the Kronecker structure making the duality exact. A system with a finite query budget can be analyzed by applying the per-query bounds to the m-fold Kronecker product; an unbounded budget needs a different approach, either rotating the trapdoor key to reset accumulated leakage or designing for the short-horizon regime where asymptotic confidentiality has not yet decayed to zero.

Adaptive adversaries

The independence the Kronecker analysis depends on fails when the stream is adaptive: the adversary chooses each query from the encrypted outputs seen so far. This is exactly the adversarial-input setting of Section 16.4, and the trapdoor paradigm inherits its failure modes, with the encrypted

outputs (not plaintext ones) driving the strategy. The bounds shift from average-case (iid Kronecker products) to worst-case: a map at $\varepsilon = 0.01$ on iid queries may reach ε near 1 on a query crafted from the preceding outputs. The response, as in chapter 16, is worst-case accounting: bound the maximum plaintext bits any adversary extracts per query and budget around that.

Consider a trapdoor membership map on a plaintext set A of $N = |A|$ records, with $\varepsilon = 2^{-k}$ and $\omega = 0$, realized by a Bloom-style construction with k keyed hash functions whose seed is the trapdoor. Under iid queries the behavior is benign: each random query has independent probability 2^{-k} of a false positive, and after m queries the server has gained $O(m)$ bits about A, spread uniformly enough that no record is pinned down. Under an adaptive adversary the picture changes, not in the per-query rate but in what a bounded budget of crafted queries buys. The $\varepsilon = 2^{-k}$ comes from k hash bits matching at once; an adversary crafts queries that probe the bits individually, each extracting about one bit of the seed, so after $O(k)$ such queries the seed is recovered and the entire dataset decrypts, and a binary search over the records localizes a chosen record's membership in $O(\log N)$ queries. The per-query extraction stays $O(1)$ bits throughout; the danger is that this bounded budget, logarithmic in the dataset size, breaks the scheme outright, where the iid attacker gains only $O(\varepsilon)$ bits per query and never recovers the seed. The gap between total compromise in a logarithmic number of crafted queries and an $O(\varepsilon)$-bit-per-query trickle is the gap between worst-case and average-case, and the fix, as chapter 16 prescribed, is a cipher map no bounded budget of crafted queries can break. Cryptographic hash families (pseudorandom functions) achieve this: a pseudorandom seed cannot be recovered by probing, so the worst-case attacker gains only one bit of best-effort guessing per query, matching the average case to a constant; the FHE sidebar that follows gives an even stronger guarantee, worst-case leakage exactly zero. Both the worst-case story (chapter 16) and the average-case story (chapters 4 through 14) are needed in the trapdoor setting: the average case for sizing under benign workloads, the worst case for security against adversarial ones.

17.4 Sidebar: Fully Homomorphic Encryption

The chapter built cipher maps as approximate functions on encrypted data, the approximate qualifier a deliberate weakening: a cipher map accepts bounded ε and ω as the price of being practical. The unweakened version is

fully homomorphic encryption, the cryptographic abstraction that promises zero-error computation on ciphertexts. This sidebar lays out what FHE is, where it sits relative to cipher maps, and when each is the right tool.

What FHE is, and what it costs

Fully homomorphic encryption (FHE) supports exact arithmetic on ciphertexts: given $\mathsf{Enc}(x)$ and $\mathsf{Enc}(y)$ it provides $\mathsf{Enc}(x) \oplus \mathsf{Enc}(y) = \mathsf{Enc}(x+y)$ and $\mathsf{Enc}(x) \otimes \mathsf{Enc}(y) = \mathsf{Enc}(x \cdot y)$, preserved through any depth. Because Boolean and arithmetic circuits decompose into additions and multiplications, FHE evaluates any computable f on encrypted inputs with zero plaintext-output error. Gentry's 2009 thesis gave the first viable construction, resolving a question open since Rivest, Adleman, and Dertouzos posed it in 1978; earlier schemes were only *partially* homomorphic (Paillier for addition, RSA and ElGamal for multiplication) and could not evaluate arbitrary circuits. For this chapter the relevant fact is that FHE realizes a cipher map with $\varepsilon = \omega = 0$: whatever uncertainty the client has after decrypting was already in the inputs, not introduced by the computation.

The zero-error guarantee is steep in three dimensions. *Ciphertext size*: an FHE ciphertext encrypts one plaintext bit but occupies kilobytes; a database fitting in gigabytes of plaintext runs to terabytes encrypted. *Operation latency*: each homomorphic operation takes milliseconds, and a lookup that costs nanoseconds on plaintext takes seconds to minutes under FHE, six to nine orders of magnitude slower. *Bootstrapping*: the ciphertext noise that accumulates with each operation must periodically be reset, and the bootstrapping that performs the reset is the most expensive FHE primitive, taking seconds even in the fastest implementations. The combined cost is why FHE has not displaced plaintext computation; it is now practical for narrow workloads (simple aggregations, fixed-circuit queries on small data) but remains too slow and too large for the broad class cipher maps address.

Cipher maps as the weaker rung

A cipher map is the strictly weaker abstraction: it admits positive ε, ω, and in exchange the designer chooses representations (hash families, sketches, Bloom filters) orders of magnitude smaller and faster than FHE's polynomial arithmetic. The framework's space-accuracy duality applies directly: a membership cipher map at $\varepsilon = 2^{-20}$ uses about $20 + \mu$ bits per element by the optimal construction of Chapter 14, against kilobytes per element for the FHE realization. FHE is the strongest and most expensive abstraction;

cipher maps are weaker but cheaper in proportion to the error tolerated, a map at $\varepsilon = \epsilon$ spending about $-\log_2 \epsilon$ bits per element of useful capacity.

When to use which, and a concrete comparison

Two questions decide it. *Does the application tolerate error?* A contract evaluation or a diagnosis lookup is exact by definition, a single bit-flip changing the meaning, so FHE is right; a top-ten search over a million-document corpus tolerates a few irrelevant results, so a cipher map at $\varepsilon = 0.01$ is right. *Does it need to scale?* A handful of exact queries per day on a small dataset runs under FHE even at minutes per query; a search index serving thousands of queries per second on multi-gigabyte data cannot, while a cipher map serves in constant time at dataset-proportional space, the same complexity as a cleartext structure. The two often combine: a cipher map as a first-stage filter rejecting most non-matches with bounded error, FHE as a second-stage verifier exactly checking the positives. Theorem 17.2.1 governs the hybrid, the FHE stage's identity channel dropping the false-positive rate to whatever it verifies.

A worked comparison fixes the ratio: set membership on $|A| = 10^6$ 64-bit integers. The optimal Bernoulli filter cipher map at $\varepsilon = 0.01$ uses about $-\log_2(0.01) \approx 6.64$ bits per element (a standard Bloom filter would spend $\log_2 e \approx 1.44$ times as much), plus a keyed seed of a few dozen bytes, for roughly 6.64×10^6 bits or about 830 kilobytes, queried in constant time. The FHE-encrypted exact set stores each element at 10^4 to 10^5 bits (1 to 10 kilobytes at 128-bit security), about 1 to 10 gigabytes total, queried by homomorphic comparisons taking seconds. The space ratio is about 10^4, the runtime ratio about 10^9, both in the cipher map's favor, the price being a one-percent false-positive rate. This is the trade-off the trapdoor paradigm exploits: FHE when error is intolerable, cipher maps for the much larger space of workloads where bounded error is acceptable. The next section turns to how Boolean combinations of cipher maps behave under the framework's composition rules.

17.5 Boolean Algebra Over Trapdoor Sets

The trapdoor cipher maps of Section 17.3 are useful in isolation and more useful combined. A common request in encrypted search is the conjunctive query, *is x in A and in B?*, asked of two trapdoor sets on the same server. The answer requires Boolean reasoning over encrypted representations, and

this section asks how Boolean algebra behaves when its operands are trapdoor sets. The framework's set-composition rules carry over, with the same three-region subtlety chapter 6 raised for plaintext approximate sets.

Trapdoor sets

Definition 17.5.1 (Trapdoor set)**.** *Let $A \subseteq X$ be a plaintext set and k a secret key. A* trapdoor set *for A under k, written $\tilde{A}$, is an encrypted data structure with a membership cipher map $\tilde{m}_A : \mathsf{Enc}_k(X) \to \mathsf{Enc}_k(\{0,1\})$ such that $\mathsf{Dec}_k(\tilde{m}_A(\mathsf{Enc}_k(x))) = \mathbf{1}[x \in A]$ with bounded rates ε_A, ω_A in the sense of Definition 17.1.1. Without k the membership query is computationally infeasible; with k it is as cheap as a plaintext lookup.*

The Bloom filter of Section 5.2 is canonical: the server holds an encrypted bit array plus key-dependent hash parameters, the client sends an encrypted query, the server returns the encrypted membership bit. The map has $\varepsilon_A = (1 - e^{-kn/m})^k$ and $\omega_A = 0$, so $\tilde{A}$ sits at $(\varepsilon_A, 0)$ on the perfect-filter line of Definition 6.4.1. A perfect-hash-filter trapdoor set (chapter 15) sits at $(2^{-r}, 0)$; a counting variant trades a small ω_A for deletions. What they share is the structure of Definition 17.5.1: encrypted representation, key-dependent query, bounded Bernoulli error.

Boolean operations and their rates

The three primitive operations lift to the trapdoor setting, the server computing on encrypted representations. *Union* returns the encrypted OR of the two membership bits (a bitwise OR of the arrays when the scheme is additively homomorphic over bits); *intersection* the encrypted AND (run both queries, combine with a homomorphic AND, since the bitwise AND of two Bloom arrays is not the filter of the intersection); *complement* the flipped bit, which by Proposition 6.3.1 moves the result to the opposite axis.

These are *parallel* combinations of two membership tests on one query, not the *sequential* chains Theorem 17.2.1 governs, so their rates are the set-composition rates of chapter 6 rather than a channel-matrix product. For two FNR-zero trapdoor sets under union, a query outside $A \cup B$ is outside both, a single region, so

$$\varepsilon_{A\cup B} \;=\; 1 - (1-\varepsilon_A)(1-\varepsilon_B), \qquad \omega_{A\cup B} \;=\; 0,$$

the chapter 3 monotone formula (Theorem 3.3.1). Intersection is subtler. A query outside $A \cap B$ falls into one of three regions, $A \setminus B$, $B \setminus A$, or outside

$A \cup B$, and the false positive arises differently in each: in $A \setminus B$ the A-test is correct and only the B-test must err (rate ε_B), in $B \setminus A$ only the A-test (rate ε_A), and only outside $A \cup B$ must both err (rate $\varepsilon_A \varepsilon_B$). By Proposition 6.2.1 the rate is the partition-weighted sum (6.5),

$$\varepsilon_{A \cap B} \;=\; w_1 \varepsilon_B + w_2 \varepsilon_A + w_3 \varepsilon_A \varepsilon_B, \qquad \omega_{A \cap B} \;=\; 0,$$

with w_1, w_2, w_3 the masses of the three regions among non-members of $A \cap B$. The bare product $\varepsilon_A \varepsilon_B$ is the $w_3 \to 1$ limit, where queries land almost entirely outside $A \cup B$; it is the best case, not the general one. The conditional independence behind the w_3 term is Axiom 1 applied to the two trapdoor sets, automatic when their keys are generated independently.

The distinction matters for the conjunctive precision often claimed for encrypted search. A k-way conjunction reaches $\prod_i \varepsilon_i$, an exponential sharpening, only in the sparse limit where queries miss every set; in general the rate is dominated by the largest single-miss region, $\max_i \varepsilon_i$ weighted by the mass of queries inside all sets but one. Deep conjunctions are highly precise on a workload of true negatives and only modestly precise on a workload of near-misses, and the design must know which regime it is in.

Cancellation escapes the perfect-filter line

The composition above kept FNR-zero operands FNR-zero, but that is not a general property: as soon as a complement enters, or operands sit on opposite axes, the result can leave the perfect-filter line, the cancellation phenomenon chapter 6 Section 6.3 flagged. Take a trapdoor union $\tilde{A} \cup \tilde{B}$ at $(\varepsilon_{A \cup B}, 0)$ and complement it: by Proposition 6.3.1 the complement of an FNR-zero set is FPR-zero, so $\complement(\tilde{A \cup B})$ sits at $(0, \varepsilon_{A \cup B})$. Now intersect with a third FNR-zero set $\tilde{C}$ at $(\varepsilon_C, 0)$. The intersection mixes axes, and the three-region rate tells the true story. Its false-negative rate is

$$\omega_{\complement(A \cup B) \cap C} \;=\; 1 - (1 - \varepsilon_{A \cup B})(1 - \omega_C) \;=\; \varepsilon_{A \cup B},$$

since $\omega_C = 0$. Its false-positive rate is *not* zero, despite one operand being FPR-zero: a query in $\complement(A \cup B) \setminus C$ (in the first operand's set, outside the second, hence outside the intersection) is a false positive whenever the C-test errs, at rate ε_C, so by (6.5)

$$\varepsilon_{\complement(A \cup B) \cap C} \;=\; w_1 (1 - \varepsilon_{A \cup B}) \varepsilon_C \;>\; 0.$$

The composed trapdoor set carries *both* a positive false-positive rate and a positive false-negative rate: it sits in the interior of the Bernoulli square,

off the perfect-filter line entirely, not merely on the opposite axis. This is the cancellation phenomenon chapter 6 documented, lifted intact to the trapdoor setting, and sharper than a naive bare-product reading suggests: mixing axes does not just swap one-sidedness but dissolves it. Designing trapdoor-set Boolean combinations requires the same care as the plaintext case, tracking operand axes and the three regions through every composition.

Worked example: conjunctive query

Take two trapdoor Bloom filters $\tilde{A}, \tilde{B}$, each at $\varepsilon = 0.01$, $\omega = 0$, and the query $q(x) = \mathbf{1}[x \in A \cap B]$. The client sends $\mathsf{Enc}_k(x)$; the server queries both, combines the encrypted bits under a homomorphic AND, and returns the result, which the client decrypts. The intersection is FNR-zero (both component queries return 1 on a true positive), and by (6.5) the false-positive rate is

$$\varepsilon_{A\cap B} \;=\; w_1(0.01) + w_2(0.01) + w_3(0.01)^2.$$

On a workload of true negatives ($w_3 \to 1$) this is the product 10^{-4}, two orders of magnitude tighter than either component, the precision gain conjunctive encrypted search is built on. On a workload where the query is known to lie in one set but tested for the other (w_1 or w_2 near 1, the natural case when A and B are correlated keyword sets), the rate is about 0.01, the single component's rate, and the conjunction buys little. The honest design figure is the weighted sum, read from the application's query distribution, not the optimistic product alone. The same set-composition law that governs plaintext Bloom-filter conjunctions governs the trapdoor lift; the next section turns the rate-algebra story into a confidentiality-algebra story, asking how much the server recovers from the conjunctive query stream.

17.6 Maximizing Confidentiality

Section 17.3 named a duality the framework has not yet quantified: the cipher-map rates ε and ω govern both correctness and confidentiality. Looser rates make the server's output less faithful to f but also less informative to an adversary inspecting it. The earlier sections treated correctness; this one treats confidentiality, and the question is concrete: given a cipher map at fixed (ε, ω) on a workload of known prevalence π, how much does the output reveal about the plaintext? The chapter 7 PPV machinery answers it with no further apparatus.

Confidentiality is a posterior

An adversary watching the server's outputs faces a posterior-inference problem: each output is a noisy report about the plaintext, and Bayes' rule combines it with prior knowledge. For a single Boolean query the target is

$$\Pr\big[\, x \in A \;\big|\; \tilde{f}(x) = \top \,\big],$$

the probability that a reported $\top$ identifies a genuine member rather than a false positive. Close to 1, the report has pinned membership down; close to the prior π, it has revealed nothing. The confidentiality is the gap, which we take to be

$$C \;=\; 1 - \mathrm{PPV},$$

the share of $\top$-reports that are false positives. Confidentiality is high when most $\top$-reports do not identify members and zero in the FHE limit, where every report is correct and membership is fully exposed.

The posterior is exactly the positive predictive value of Definition 7.4.1. Reading the chapter 7 formula in the cipher-map setting,

$$\mathrm{PPV} = \frac{\pi(1-\omega)}{\pi(1-\omega) + (1-\pi)\varepsilon}, \qquad C = 1 - \mathrm{PPV} = \frac{(1-\pi)\varepsilon}{\pi(1-\omega) + (1-\pi)\varepsilon},$$

the numerator $\pi(1-\omega)$ the rate of correct $\top$ on a member, $(1-\pi)\varepsilon$ the rate of spurious $\top$ on a non-member, C the share that protects the plaintext. Zero false positives gives $C = 0$; an unconditional $\top$ gives $C = 1 - \pi$. The negative case mirrors it, the confidentiality of a $\bot$-report being $1 - \mathrm{NPV}$ (Definition 7.4.2), though in typical encrypted search π is small and the $\top$-side ambiguity dominates.

The $C = 1-\mathrm{PPV}$ measure is the per-report complement; the information-theoretic complement, in bits, is also available.

Remark (Confidentiality in bits per query)**.** *Model the adversary's per-query posterior as Bernoulli at* PPV *for a* $\top$*-report and at* $1 - \mathrm{NPV}$ *for a* $\bot$*-report. The leakage is the mutual information between the plaintext bit and the report,*

$$L \;=\; H(\pi) - \mathbb{E}_{report}[H(\textit{posterior})] \;=\; H(\pi) - \big[\Pr(\top)\, H(\mathrm{PPV}) + \Pr(\bot)\, H(1-\mathrm{NPV})\big],$$

with H *the binary entropy and* $\Pr(\top) = \pi(1-\omega) + (1-\pi)\varepsilon$*. In the FHE limit* $\varepsilon = \omega = 0$ *the posteriors are deterministic,* $L = H(\pi)$*, the full prior entropy leaks. A cipher map with positive rates leaks a fraction of* $H(\pi)$ *per query, the information-theoretic counterpart of the* $C = 1 - \mathrm{PPV}$ *posterior, and the chapter 8 entropy machinery measures that fraction directly.* △

Trading rate-budget for confidentiality

On the perfect-filter line of Definition 6.4.1 the designer faces a one-dimensional trade-off. An $\omega = 0$ map sits on the ε-axis, and raising ε raises $(1 - \pi)\varepsilon$ and so raises C monotonically. Walking the line away from the origin trades correctness for confidentiality. The trade is not free in the convenient direction: chapter 5's space formula makes storage a decreasing function of ε, so *lower-storage cipher maps are more confidential*, one of the framework's small surprises, following entirely from the chapter 7 posterior with no cryptographic argument. Storage couples to accuracy through the chapter 8 entropy bound and accuracy to confidentiality through the chapter 7 posterior, so one parameter sets three coupled costs. Maximizing C in the abstract is degenerate (walk to $\varepsilon = 1$, $C = 1 - \pi$, the map useless), so the real problem constrains the operational error: maximize C subject to $\varepsilon \le \varepsilon^*$ for an application-imposed cap, the optimum being the cap itself.

A worked example

Take prevalence $\pi = 0.01$ and rates $\varepsilon = 0.05$, $\omega = 0$. The PPV formula gives

$$\mathrm{PPV} = \frac{0.01}{0.01 + 0.99 \cdot 0.05} = \frac{0.01}{0.0595} \approx 0.168, \qquad C \approx 0.832.$$

When the server reports $x \in A$, the adversary's posterior assigns about 16.8% to genuine membership and 83.2% to a false positive; one query leaves the question far from settled. The FHE baseline reports with zero error, PPV $= 1$, $C = 0$: every report is correct and fully revealing. The contrast is stark, the cipher map paying a 5% false-positive rate for an 83% confidentiality margin, FHE paying a $10^4\times$ space and $10^9\times$ time overhead for a zero margin. The example also shows the workload dependence: holding $(\varepsilon, \omega) = (0.05, 0)$ but moving to a balanced workload $\pi = 0.5$ gives $\mathrm{PPV} = 0.5/0.525 \approx 0.952$ and $C \approx 0.048$. The same map is confidential on a rare-positive workload and revealing on a balanced one. The framework keeps the parameters separate: the map carries (ε, ω), the workload carries π, and $C(\varepsilon, \omega, \pi)$ is one function the designer plots and reads. Chapter 7 did the algebra; chapter 17 inherits it.

17.7 Bridge

The chapter built one bridge in six pieces. The cipher-map abstraction (Section 17.1) reframed encryption as a pair of error rates rather than a pair

of keys; the error-model section (Section 17.2) identified those rates with the Bernoulli atom and lifted the Kronecker composition law to chained cipher-map evaluation; the trapdoor-paradigm section (Section 17.3) introduced the client-server split, and the FHE sidebar (Section 17.4) marked the zero-error baseline. Section 17.5 and Section 17.6 pushed the framework forward in two directions, Boolean composition over trapdoor sets and confidentiality as a chapter 7 PPV.

The dual role of the error rates

The recurring thread is the dual role of ε and ω. In Parts I through IV they were operational parameters, kept small subject to storage and runtime budgets. Here they acquire a second job: the false-positive rate is also the share of an adversary's $\top$-observations that fail to identify members, the false-negative rate the share of $\bot$-observations that protect real members, and together they fix the map's PPV at prevalence π, the complementary $1 - \mathrm{PPV}$ being the confidentiality margin. The duality is not a coincidence. The framework identifies a structure by its (ε, ω) point; chapter 6 read that point geometrically as a position on the perfect-filter line, chapter 7 predictively as PPV and NPV, and chapter 17 cryptographically as the trade-off between operational fidelity and informational opacity. The three readings are the same algebra, and a single choice of where to live in the Bernoulli square now answers three questions at once.

Forward to chapter 18

Chapter 18 specializes the machinery to encrypted search, fixing f to a membership-test family, "does record r match query q?", and asking how a server holding an encrypted index answers faithfully without revealing r or q. The Bloom-filter cipher maps of Section 17.5 reappear as the simplest index, the chapter 14 BHF as a space-optimal alternative, and the two-party setup of Section 17.3 stays in force, the (ε, ω) point still doing triple duty. What chapter 18 adds is the structure the encrypted-search literature has built around the membership case: data-oblivious access patterns, side-channel control, multi-query leakage bounds, and the relation between the confidentiality C and the information-theoretic measures that literature prefers. Chapter 17 is the abstract algebra; chapter 18 is the concrete instance, the same move chapters 14 and 15 made together. Section 17.7 attaches the chapter's prose to its antecedents in the FHE, cipher-map, and secure-computation literatures.

Bibliographic Notes

The cipher-maps manuscript. The cipher-map abstraction of Section 17.1 is developed at length in Towell [Tow26f], "Cipher maps: a unified framework for oblivious function evaluation," which the present chapter summarizes. The manuscript treats the abstraction in its own right rather than as a special case of the Bernoulli framework, classifies cipher-map families by their channel-matrix structure, and develops the cryptographic-side analysis (key schedules, side-channel considerations, hardness assumptions) the chapter has not entered. The chapter's contribution beyond the manuscript is the identification of cipher-map rates with Bernoulli atoms and the lifting of chapter 4's Kronecker composition to cipher-map composition, which the manuscript treats algebraically without anchoring to the Bernoulli square.

The trapdoor-computing series. The Boolean-algebra section (Section 17.5) draws on Towell [Tow26e], "Boolean algebra over trapdoor sets," which gives the full treatment of union, intersection, complement, and conditional operations over trapdoor representations, including the cancellation phenomena and the three-region composition rates the present chapter uses. The maximizing-confidentiality section (Section 17.6) draws on Towell [Tow26i], "Maximizing confidentiality in encrypted search," which develops the PPV-as-confidentiality connection and extends it to information-theoretic measures (observed-versus-maximum entropy ratios, compression-based estimation) the chapter has only gestured at in Section 17.6. A third strand, on algebraic cipher types, generalizes the set-and-map treatment to algebraic data types and remains in active drafting.

Fully homomorphic encryption. The FHE sidebar of Section 17.4 stands on Gentry [Gen09], "Fully Homomorphic Encryption Using Ideal Lattices," the first plausibly-secure fully homomorphic scheme, combining a somewhat-homomorphic scheme with a bootstrapping construction that refreshes ciphertexts faster than homomorphic noise accumulates. Modern schemes (BFV, BGV, CKKS) cut the per-operation cost by orders of magnitude but inherit the bootstrapping idea and remain several orders slower than the cipher-map approach. The standard FHE references are surveys rather than primary sources; the reader should follow the literature trail from Gentry's paper.

The secure-computation tradition. The trapdoor-paradigm framing of Section 17.3 sits in a literature going back to early work on secure com-

putation. Goldreich [Gol96], "Foundations of Cryptography: Fragments of a Book" (cited in chapter 16 for the oblivious-Turing-machine model), gives the foundational treatment of computation under cryptographic constraints, including the relation between secure computation and the multiparty setting that cipher maps generalize. Naor and Yung [NY89], "Universal One-Way Hash Functions and their Cryptographic Applications," is the older entry point to the trapdoor-function line the cipher-map abstraction inherits its name from. The chapter has used "trapdoor" in the colloquial sense rather than the strict cryptographic one, but the connection to the formal trapdoor-function literature is real, and the encrypted-search applications of chapter 18 activate it more explicitly. The secure-multiparty-computation literature is enormous; this treatment is necessarily brief.

Chapter 18

Encrypted Search as Relaxed Trapdoor

18.1 The Encrypted Search Problem

A client stores documents on a remote server and wants to search them for a keyword without trusting the server to read either the documents or the queries. The documents hold private content, the queries reveal what the client is interested in, and handing either to the server in the clear defeats the point. The server has abundant storage and computation and will do work on the client's behalf; what it cannot be trusted with is reading the data it stores.

Definition 18.1.1 (Encrypted-search problem)**.** *A client holds a corpus of plaintext documents $\{d_1, \ldots, d_n\}$, each a set of keywords from a vocabulary W. The client wants to outsource the corpus to a server so that, for any keyword $w \in W$, it can issue a query the server resolves by returning the identifiers $\{i : w \in d_i\}$, while the server learns neither the plaintext of any document nor the keyword of any query.*

The two privacy goals are intertwined: whatever operation tests a query against a stored document is the seam through which information leaks. A scheme leaking nothing about either side does not exist in general, since the server must learn something to return an answer, if only the bare fact that a query matched. The design question is how much, and of what kind.

Three naive resolutions

The first resolution skips the server: download everything and search locally. It leaks nothing but eliminates every reason to use a server, since a client that can download and store the full corpus could have hosted it locally. The second preserves the server at the cost of privacy: send queries in plaintext and let the server match them. If the documents are plaintext the server reads everything; if they are encrypted but the queries are not, the server still learns the workload (which keywords matter, how often, in what order) and correlates it with access patterns, timing, and document sizes to recover content in many cases. Encrypting the documents but not the queries protects the corpus only as long as the workload is uninformative, and most realistic workloads are extremely informative. The third chases full privacy through fully homomorphic encryption (Section 17.4), which lets the server evaluate an arbitrary circuit on encrypted inputs but is many orders of magnitude too slow for the corpus scales encrypted search must address. The first abandons the server, the second privacy, the third performance. The fourth resolution, this chapter's, abandons *exactness*.

Encrypted search as a cipher map

The cipher map of Definition 17.1.1 operates on encrypted representations and returns the right answer up to bounded ε and ω, and encrypted search fits it directly. The plaintext function is the membership test "does document d contain keyword w?"; the encrypted-domain map is a server-side test on whatever encrypted form of d and w the construction provides; the error rates are the share of non-members reported as matches and the share of true members missed. A search is a sequence of such tests, one per document, and the Kronecker composition law of Section 17.2 aggregates the per-test rates into a corpus-wide result. Encrypted search is thus a family of cipher maps parameterized by the underlying approximate-set structure and the encryption scheme applied to it; the two rates govern recovery (how faithfully the response reflects the true match set) and the confidentiality of Section 17.6 governs leakage (how much the response reveals about the plaintext). The Bernoulli machinery applies unchanged; only the operational reading of ε, ω changes.

The leakage-recovery trade-off

A tighter system (smaller ε, ω) returns a more faithful match set but also leaks more: its membership reports are more reliable, so a third party who

learns a report has a higher posterior on whether the document truly contained the queried keyword. The positive predictive value of Definition 7.4.1, $\Pr[w \in d \mid \text{match reported}]$, read with the adversary in mind, is the leakage measure, and confidentiality is $1 - \text{PPV}$ (Section 17.6), the two halves of the trade-off being opposite faces of one formula. The trade-off is unavoidable: an exact system has $\text{PPV} = 1$ and confidentiality zero, a system returning random matches has confidentiality $1 - \pi$ and is useless, and every construction picks a point between. Section 18.2 builds two Bloom-filter constructions, Section 18.3 works out the PPV/NPV/F1 measures as confidentiality bounds, Section 18.4 addresses estimating leakage under adversarial workloads, and Section 18.5 walks an end-to-end implementation.

18.2 Bloom-Filter-Based Constructions

The cipher-map framing of Section 18.1 identified encrypted search with a family of approximate membership tests on encrypted documents. A construction needs an approximate-set structure recording each document's keywords compactly, an encryption scheme letting the server test a query token against the stored representation without learning the plaintext keyword, and a query-token format the client computes privately and the server matches obliviously. The Bloom filter of Definition 5.2.1 solves the first; a standard symmetric scheme (a one-time pad keyed by a pseudorandom function) the second; a deterministic keyed function of the keyword the third. The two construction families differ in what counts as a "document" in the Bloom sense: the first puts a filter inside each document, the second inside each keyword.

The two constructions

Definition 18.2.1 (Per-document Bloom-filter encrypted search)**.** *For each document d_i the client computes a Bloom filter B_i over its keywords, encrypts it as $\mathsf{Enc}(B_i)$ under a key the server does not hold, and uploads $\{(i, \mathsf{Enc}(B_i))\}$. To search for w the client sends a query token τ_w (a deterministic function of w and its secret key); the server tests τ_w against each $\mathsf{Enc}(B_i)$ and returns the identifiers for which the test succeeds.*

Definition 18.2.2 (Inverted-index Bloom-filter encrypted search)**.** *For each keyword w the client computes a Bloom filter B_w over the identifiers $\{i : w \in d_i\}$, encrypts it, and uploads $\{(\tau_w, \mathsf{Enc}(B_w))\}$ indexed by token. To search*

for w the client sends τ_w; the server returns the matching encrypted filter, which the client decrypts to recover an approximate identifier set.

The per-document construction asks, for each document, "does this query match?", and has $|D|$ answers per query; the inverted-index construction asks "which documents match?", with $|W|$ index entries and a shorter query path (one lookup, one encrypted blob, the rest done client-side). The encryption layer does two jobs in both: it hides the bit pattern from the server, and it makes the match oblivious to the underlying keywords, since τ_w carries w only through a keyed function the server cannot invert.

Each is a cipher map, and FNR-zero

In the per-document construction the plaintext function is $f_i(w) = \mathbf{1}[w \in d_i]$: the token replaces w, the encrypted filter replaces d_i, and the server-side match approximates f_i up to the filter's ε, always reporting a match when $w \in d_i$ ($\omega = 0$) and reporting one with probability at most ε when $w \notin d_i$. The error profile is $(\varepsilon, 0)$, satisfying Definition 17.1.1. The inverted-index construction realizes $f(w) = \{i : w \in d_i\}$ with the same profile. Both have $\omega = 0$ by structural inheritance from the Bloom filter, whose union analysis (Section 6.1) showed that ORing bits can only add matches, never remove them, so the per-document filter never misses an inserted keyword.

The chapter 3 composition theorem (Theorem 3.3.1) extends this to multi-keyword conjunctions, with the three-region care chapter 6 required for intersection. A client querying for documents containing both w_1 and w_2 intersects the two returned identifier sets. The result has $\omega = 0$ (a true match, containing both keywords, is returned by both queries and survives). Its false-positive rate per document is bounded by ε, but not for the reason a bare product would suggest. A document outside the conjunction is a false positive when the test for a keyword it lacks falsely matches: a document containing w_1 but not w_2 is returned whenever the w_2-test errs, at rate ε (the w_1-test is correct, not a false match), and only a document containing neither keyword needs both tests to err, at the lower rate $\approx \varepsilon^2$. By the three-region intersection rate (Proposition 6.2.1, (6.5)) the conjunction's FPR is the partition-weighted sum, bounded by ε and met by documents missing exactly one of the two keywords. Deep conjunctions are highly precise on a corpus of true non-matches and only modestly precise when documents commonly contain one queried keyword but not the other.

A Bloom-based encrypted search therefore never *loses* a real match: every document truly containing the keyword appears. What varies is how

many *additional* documents appear as false positives, the operational price the client pays, exactly the price chapter 5 set for the underlying filter.

Space and query cost

For the per-document construction with $|D|$ documents and W_d keywords each, the chapter 5 optimal-k formula (Theorem 5.2.1) gives $m_d/W_d = (\log_2 e)\log_2(1/\varepsilon) \approx 1.44\log_2(1/\varepsilon)$ bits per item, so total storage is

$$S_{\text{per-doc}} \;\approx\; 1.44 \cdot |D| \cdot W_d \cdot \log_2(1/\varepsilon),$$

at $|D|$ server-side matches per query. The inverted-index construction with vocabulary $|W|$ and average per-keyword document count $|D_w|$ costs $S_{\text{inv}} \approx 1.44 \cdot |W| \cdot |D_w| \cdot \log_2(1/\varepsilon)$ at one server lookup per query. Since $\sum_d W_d = \sum_w |D_w|$ is the total keyword-document incidence, the two cost the same in storage to leading order; they differ in query cost by a factor of $|D|$ at the server (the per-document construction tests every filter, the inverted-index does one lookup), at the price of pushing the membership test to the client. Incremental update favors the per-document layout (adding a document is local) while query latency favors the inverted index, so production deployments routinely keep both, the per-document filters as the primary store and the inverted index as a query accelerator rebuilt lazily.

A worked example

Take $|D| = 10{,}000$ documents, $W_d = 50$ keywords each, $\varepsilon = 0.01$. The per-document filter has $1.44\log_2(100) \approx 9.59$ bits per item, so $m_d \approx 50 \cdot 9.59 \approx 480$ bits, total $10{,}000 \cdot 480 = 4.8 \times 10^6$ bits $= 0.6$ MB. Exact word-set storage at 50 keywords of 8 bytes each is $10{,}000 \cdot 50 \cdot 8 = 4$ MB, about seven times more, and the Bloom representation also encrypts a fixed-size object regardless of keyword length. The inverted-index version needs the same $\sum_d W_d = 5 \times 10^5$ incidence pairs at 9.59 bits each, the same 0.6 MB: the construction style changes the layout and the query path, not the storage cost. At $\varepsilon = 0.01$ over this corpus the expected spurious matches per query are $0.01 \cdot |D| = 100$, against perhaps $|D| \cdot \pi = 10$ true matches at prevalence $\pi = 0.001$; the response is dominated by false positives, which the client filters out by some means external to the encrypted-search layer. The next section reads those false positives as the source of the system's confidentiality rather than as a defect.

18.3 Confidentiality Measures

The cipher-map framing of Section 18.2 fixed an error profile $(\varepsilon, 0)$ for both constructions, and Section 17.6 worked out confidentiality at fixed rates and prevalence. This section reads that derivation in the encrypted-search setting: every chapter 7 measure has a leakage interpretation, and trading rate-budget for confidentiality reduces to choosing ε on the chapter 7 curve.

The server's leakage model

A deployed system gives the server two streams. The *structural* stream is the encrypted filter store, opaque by construction: each $\mathsf{Enc}(B_i)$ is semantically secure, leaking nothing beyond the filter size. The *operational* stream is where the analysis lives: each match decision is a yes/no answer to "does τ_w correspond to a keyword in d_i?", which the server computes and therefore knows. The plaintext keyword and document content stay hidden, but the bare fact of the match is unavoidably revealed. Under the Bernoulli model each match decision is a noisy bit, 1 when the keyword truly belongs (the filter's $\omega = 0$) and 1 with probability ε when it does not, so the chapter 7 measures over noisy bits are the right measures for the server's posterior.

PPV is the confidentiality measure

Fix a query token τ_w and a document d_i whose filter matched. The server's posterior that w truly appears in d_i is, by Bayes' rule, the positive predictive value of Definition 7.4.1 at the cipher map's rates and the workload prevalence π,

$$\Pr[w \in d_i \mid \tau_w \text{ matched}] = \mathrm{PPV} = \frac{\pi(1-\omega)}{\pi(1-\omega) + (1-\pi)\varepsilon}.$$

The chapter 7 formula and the cipher map's posterior are the same quantity derived twice. Following Section 17.6, confidentiality is the complement $C = 1 - \mathrm{PPV}$, high when most matches are spurious and low when most are genuine. At $\omega = 0$, which both constructions satisfy,

$$C \;=\; \frac{(1-\pi)\varepsilon}{\pi + (1-\pi)\varepsilon},$$

the single most useful equation in the chapter: on the perfect-filter line confidentiality is monotone in the rate-prevalence product $(1-\pi)\varepsilon$ relative to π. The companion measure for $\perp$-reports is the negative predictive value

(Definition 7.4.2), which at $\omega = 0$ collapses to NPV = 1: a Bloom-based search reports $\perp$ only when the keyword is truly absent (the union analysis of Section 6.1), so a $\perp$-report is fully reliable and the $\omega = 0$ regime concentrates all confidentiality on the $\top$-side. The balanced-workload summary is the F_1-score of Definition 7.4.3, which at $\omega = 0$ (recall = 1, precision = PPV) reduces to $2\mathrm{PPV}/(1+\mathrm{PPV})$, monotone in PPV and the right measure when π is not known in advance. The chapter 7 apparatus (Section 7.4) thus carries over unchanged, every derived measure acquiring a confidentiality reading.

Multi-query leakage

A single query leaks a bounded amount about one (w, d_i) pair; a long sequence leaks about the whole corpus, governed by the Kronecker structure of Section 4.4 that Section 17.3 already used.

Proposition 18.3.1 (Multi-query leakage scaling)**.** *For m independent queries against a system with profile $(\varepsilon, 0)$ and prevalence π, under the chapter 4 independence axiom the server's posterior factorizes as the m-fold Kronecker product of single-query posteriors. The total leakage (mutual information between the query stream and the plaintext) grows linearly, $I_m = m\, I_1$, while the confidentiality of any single (w, d_i) pair, measured in bits of posterior uncertainty, shrinks as $C_m = \Theta(m^{-1/2})$ by standard concentration.*

The two scalings read the same phenomenon from opposite sides: the server's total information grows linearly in the budget, while the confidentiality of any one fact decays as its square root, the asymmetry coming from the central limit theorem sharpening the posterior mean at rate $m^{-1/2}$. A system with a fixed query budget can be analyzed by reading off C_m; an unbounded budget cannot have non-trivial asymptotic confidentiality, so the trapdoor key must be rotated periodically (a rotation resets the accumulated leakage) or the asymptotic posterior must itself be acceptable.

A worked example

Take the per-document construction (Definition 18.2.1) at $\varepsilon = 0.01$, $\pi = 0.1$ (one queried keyword in ten truly appears in the queried document). The PPV formula at $\omega = 0$ gives

$$\mathrm{PPV} = \frac{0.1}{0.1 + 0.9 \cdot 0.01} = \frac{0.1}{0.109} \approx 0.917, \qquad C \approx 0.083,$$

a correctness-tuned, privacy-leaky point: a match gives the adversary a 91.7% posterior. Loosening to $\varepsilon = 0.3$ gives PPV $= 0.1/0.37 \approx 0.270$, $C \approx 0.730$, the matches now mostly false positives and the plaintext well protected, at the cost of a much noisier response the client must post-filter. Every ε at fixed π sits on the curve $C = (1-\pi)\varepsilon/[\pi + (1-\pi)\varepsilon]$; intermediate points are $\varepsilon = 0.05$ at $C \approx 0.310$, $\varepsilon = 0.1$ at $C \approx 0.474$, $\varepsilon = 0.2$ at $C \approx 0.643$. Inverting for the design target, $\varepsilon^* = \pi C/[(1-\pi)(1-C)]$, so $C = 0.9$ at $\pi = 0.1$ needs $\varepsilon^* = 1$ (degenerate), while the same target at $\pi = 0.01$ needs $\varepsilon^* \approx 0.091$, a friendly operating point: lower-prevalence workloads buy confidentiality more cheaply, matching the chapter 17 intuition that confidentiality is abundant where the prior already concentrates the plaintext on $\perp$. The next section turns to the harder problem of estimating π when the workload is adversarial.

18.4 Estimation under Adversarial Queries

Section 18.3 priced confidentiality from ε, ω, and the workload prevalence π. The first two are fixed at deployment; the third is a property of how the system is used, and a defender who wants the actual leakage must estimate π from live traffic. The problem looks elementary, each query matching or not, the matches Bernoulli, the chapter 7 machinery pricing a confidence interval. The complication is the source of the queries. A benign iid workload is the textbook setting of Section 7.2, and the Wilson and Clopper-Pearson intervals of Section 7.3 deliver a CI in one line. An adversarial workload, where the attacker chooses each query to maximize leakage and adapts to responses, is neither iid nor stationary: the per-query match probability is a function of the attacker's strategy, not a fixed parameter. The iid CI then bounds the empirical match rate the defender just measured, not the worst-case rate the attacker can sustain, the wrong quantity to feed the confidentiality formula.

Multi-armed bandit framing

The right framing treats each distinct keyword the attacker might query as an arm in a multi-armed bandit: each pull is one query, the reward is the leakage it produces, and the defender's goal is not to maximize reward but to bound the worst arm's, a high-confidence statement that no keyword exceeds a target leakage budget. The upper-confidence-bound family maps directly: the defender maintains a per-arm estimate $\hat{\pi}_w$ and the UCB $\hat{\pi}_w + \sqrt{2\log(1/\delta)/n_w}$, a high-confidence upper bound that is *adversarially valid*,

holding over all arms and all attacker strategies because it does not assume the arm-selection rule is iid. The maximum per-arm UCB is the worst-case bound fed into the confidentiality formula. Thompson sampling is the Bayesian counterpart, a Beta(α_w, β_w) posterior per arm with the maximum sampled rate reported; both agree that the worst arm is what must be bounded.

Confidence intervals under adversarial sampling

The Wilson and Clopper-Pearson CIs adapt by a union bound over arms. For a vocabulary of K arms at joint confidence $1 - \delta$, computing each per-arm CI at $1 - \delta/K$ and taking the maximum is valid by Bonferroni. The price is a $\log K$ factor in the per-arm width: the Gaussian quantile rises from about 1.96 at the marginal $\delta = 0.05$ to $\sqrt{2\ln(K/\delta)}$ under the correction, which at $K = 10^5$ and $\delta = 0.05$ is $\sqrt{2\ln(2 \times 10^6)} \approx \sqrt{29} \approx 5.4$, widening each interval by roughly a factor of 2.7. A tighter alternative is the construction Towell [Tow16] developed for this setting: the defender estimates not K independent quantities but a single one, the worst-case confidentiality as the maximum of K rates, and applies an order-statistic-aware moving-average bootstrap, resampling the empirical trace with a moving window and reporting the worst-window rate with the bootstrap quantile as the tail probability. The bootstrap CI is narrower than the union bound when the per-arm rates are similar and degenerates to it when one arm dominates. The defender chooses the construction from what is known of the attacker: the union bound under a worst-case assumption, the bootstrap when the attacker concentrates on a few keywords.

Why the iid bound is the wrong bound

The chapter 11 multiplicative Chernoff bound (Theorem 11.4.1) sharpens the iid CI to exponential decay, the right bound for iid traffic and the wrong one for adversarial traffic, because its derivation invokes the iid moment generating function, which assumes the arm-selection rule is state-independent. The UCB bound replaces that MGF with an adversarial concentration argument surviving state-dependent selection; the two agree on iid traffic (the UCB constant $\sqrt{2\log(1/\delta)}$ matches the Chernoff exponent for moderate δ) and diverge on adversarial traffic, where Chernoff under-bounds the true rate and UCB over-bounds it correctly. The chapter 16 boundary discussion (Section 16.4) reads here as a constructive recipe: the Bernoulli model itself is correct (the per-query match indicator is still Bernoulli given the

queried keyword); what changes is the estimator aggregating the indicators into a confidentiality bound.

A worked example

A defender monitors a system at 100 queries per minute, processing $n = 6000$ queries in an hour and observing 540 matches, an empirical rate $\hat{\pi} = 0.09$. The *iid* normal-approximation interval at 95% is $0.09 \pm 1.96\sqrt{0.09 \cdot 0.91/6000} \approx 0.09 \pm 0.0072$, so $\pi \in [0.083, 0.097]$; substituting the upper end into the confidentiality formula at $\varepsilon = 0.01$, $\omega = 0$ gives PPV ≈ 0.915, $C \approx 0.085$, so the defender concludes confidentiality is at least 0.085. The *MAB UCB* approach, with $K = 1000$ keywords queried over the hour at $\delta_{\text{arm}} = 0.05/1000 = 5 \times 10^{-5}$ and an average $n_w = 6$ queries per arm (with high variance), bounds an arm at $n_w = 6$, $\hat{\pi}_w = 0.5$ by a Clopper-Pearson upper limit near 1 (about 0.99 at this sample size and joint confidence); the worst-arm bound is then $\pi_{\max}^{\text{MAB}} \approx 0.99$, giving PPV ≈ 0.9999 and $C \approx 10^{-4}$. The two bounds differ by nearly three orders of magnitude, and only the MAB bound answers the defender's actual question, how leaky is the worst-case keyword the attacker might target. The warning is the chapter's recurring one: a framework guarantee is only as strong as the estimator that produced its parameters, and the iid estimator misapplied to adversarial data, not the Bernoulli machinery, is at fault. The next section turns from estimation back to construction.

18.5 A Toy Encrypted Search System

This section turns the framing into the smallest running form that exhibits every piece of the framework: a single-client, single-server, per-document Bloom system with a symmetric cipher for the index and a keyed function for the query tokens. It is deliberately minimal, handling no multi-user key management, no index rotation, and no side-channel defense beyond the primitives' standard guarantees; what it does is show the cipher-map abstraction in operational form, the client encrypting something, the server operating on the ciphertext to return an approximate answer, and the framework pricing the recovery-leakage trade-off at every stage. The companion notebook materializes it; the prose gives the structure the notebook fills.

Construction, setup, and query

Three cryptographic ingredients suffice: a keyed function HMAC_k producing a query token $\tau_w = \mathrm{HMAC}_k(w)$, deterministic in w and pseudorandom to anyone without k; a symmetric authenticated encryption (AES-based) for the bit arrays; and the Bloom filter of Definition 5.2.1 as the per-document structure. Each document's keyword set becomes a filter B_i, encrypted as $\mathsf{Enc}(B_i)$ and uploaded with its identifier; the client keeps both keys, the server keeps the encrypted store, and neither party holds both the plaintext keywords and the bits they hash to. The match operation is the cipher map: to test whether τ_w corresponds to a keyword in d_i, the server checks whether the bit positions τ_w addresses are all set in the recovered filter, reporting $\top$ ("d_i may contain the keyword") or $\bot$ ("d_i definitely does not," by $\omega = 0$). The simplest variant decrypts the filter server-side, a security regression that production variants avoid with bilinear pairings or one-time-pad-keyed arrays at extra cost; for the toy system the simplification is acceptable.

Setup runs once per corpus on the client: for each document, extract the keyword set, build a filter at the target ε (the chapter 5 size $m \approx 1.44\, W_d \log_2(1/\varepsilon)$), encrypt, upload. A query is a two-round-trip protocol: the client computes τ_w locally and sends it, the server matches it against every encrypted filter and returns the matched identifiers, and the client verifies the candidates (by a second-pass decryption outside the encrypted-search layer) and discards the false positives. The server never sees the plaintext keyword (the HMAC hides it), the plaintext keyword set (the filter hides it up to false positives), or the verification step (entirely client-side). What it sees is the cipher-map output stream chapter 17 predicted, which the chapter 7 PPV prices.

Trace through a single query

Query `alice` against $|D| = 1000$ documents at $\varepsilon = 0.01$, $\omega = 0$, with true match set $\{17, 209, 832\}$. The client computes τ_{alice}, a fixed-size pseudorandom string; the rates enter through the filter parameters fixed at setup, and the prevalence is $\pi = 3/1000 = 0.003$. The server tests the token against all 1000 filters: the three true matches return $\top$ unconditionally ($\omega = 0$), and the 997 non-matches return $\top$ at rate 0.01, an expected 9.97 false positives, for a reported set of about 13. The client decrypts the candidates, confirms the three, and discards the rest. The query's PPV is

$$\mathrm{PPV} = \frac{0.003}{0.003 + 0.997 \cdot 0.01} \approx \frac{0.003}{0.01297} \approx 0.231, \qquad C \approx 0.769,$$

the matches dominated by false positives and the confidentiality margin correspondingly large. The trace exhibits every framework parameter in operational form, each predicted before the trace ran.

Tuning for privacy

The toy system as built is correctness-tuned. A production trace at a higher prevalence $\pi = 0.1$ (queried keywords are *interesting* keywords) with $\varepsilon = 0.01$ gives PPV ≈ 0.917, $C \approx 0.083$: a server compromise exposes most of the plaintext, an adversary inferring the keyword at a 92% posterior. Retuning is mechanical: solving the chapter 7 PPV equation, $\varepsilon^* = \pi C/[(1-\pi)(1-C)]$, a target $C = 0.5$ at $\pi = 0.1$ needs $\varepsilon^* = 0.05/0.45 \approx 0.111$. The filter shrinks correspondingly, $m \approx 1.44 \cdot 10 \cdot \log_2(9) \approx 46$ bits against 96 at $\varepsilon = 0.01$, and the client's post-filtering load grows. No choice escapes the trade-off; the only design freedom is which corner of the space-accuracy-confidentiality space to occupy.

Remark (The framework as a design language)**.** *The three budgets of an encrypted-search deployment each have a price the framework reads off a different chapter. A* privacy *budget, a target confidentiality C^*, fixes the false-positive rate through the chapter 7 posterior, $\varepsilon \geq \pi C^*/[(1-\pi)(1-C^*)]$. A* correctness *budget, a cap B on expected false positives per query over a corpus of $|D|$ documents, fixes it from the other side, $\varepsilon \leq B/|D|$. A* storage *budget caps the bits through the chapter 5 and chapter 8 space law, $1.44\,|D|\,W_d \log_2(1/\varepsilon) \leq S$, a lower bound on ε. The three constraints carve a feasible interval for ε at a given π, and the design rule reads the operating point off it, or reports that no feasible point exists when the privacy floor exceeds the correctness ceiling, as happens when a high C^* is demanded at a high prevalence (the $\varepsilon^* = 1$ degeneracy of Section 18.3). The same triangle priced chapters 6, 7, 8, 14, and 17; the encrypted-search designer solves it as one system of inequalities.* △

Notebook hook

The companion notebook, `bernoulli-py/notebooks/ch18-encrypted-search.ipynb`, implements the toy system end-to-end: per-document filters, AES-encrypted bit arrays, HMAC-based tokens, server-side matching, and verification against ground truth. It plots empirical PPV and confidentiality across ε from 0.001 to 0.5, exhibiting the trade-off curve as a measured quantity, and runs the Section 18.4 comparison between iid Chernoff and MAB UCB intervals on a synthetic adversarial trace, reproducing the two-orders-of-magnitude gap.

It is a starting point, not a production artifact; a real deployment should consult the SSE literature and the primitives' formal security definitions, which the notebook does not engage.

18.6 Closing the Book

The book built one framework in five parts and eighteen chapters, each part lifting the preceding part's algebra to a larger domain and each chapter filling in one component of the lift. This section traces the arc.

The atom and its lifts

Part I opened with the noisy bit (Chapter 1), a Boolean value crossing a binary channel that flips it with known probability, and named the two atomic parameters, the false-positive rate ε and the false-negative rate ω, jointly inhabiting the Bernoulli square. Chapter 4 formalized the atom: Bernoulli[bool] (Definition 4.5.1) collects all (ε, ω) channels, the two axioms (Axiom 1, Axiom 2) prescribe how atoms combine, and the Kronecker theorem (Theorem 4.4.1) gives the composition law as a matrix product. Part I closed with the whole machinery distilled into one 2×2 matrix per atom and a Kronecker product for composition.

Part II lifted the atom to a set of bits, one per element of a universe. Chapter 5 framed the random approximate set as a function from a plaintext set to a sequence of atoms; Chapter 6 worked out the composition theorem for union, intersection, complement, and difference, with the perfect-filter line marking the structures (Bloom filters chief among them) on the $\omega = 0$ axis; Chapter 7 lifted the atomic rates to the derived measures (precision, recall, PPV, NPV, F_1, Youden's J); and Chapter 8 priced approximate membership at the entropy floor $-\log_2 \varepsilon$ bits per element. Part II closed with the algebra of approximate sets fully priced: construction bounded by Shannon, composition by Kronecker, classification by the chapter 7 posterior.

Part III lifted from sets to functions. Chapter 9 took the binary channel to its n-ary generalization, the random approximate map; Chapter 10 treated the streaming sketches that compress the map into bounded memory; Chapter 11 extended the framework to randomized algorithms; Chapter 12 took the relational lift; and Chapter 13 unified them, every Bernoulli-lifted structure an instance of Bernoulli[T] for some type T, the compositions inheriting from the algebra of types.

The optimum, the boundary, the applications

Part IV asked whether the space-accuracy lower bound is achievable. Chapter 14 answered yes for the FNR-zero set: the Bernoulli hash function reaches the chapter 8 entropy floor within a vanishing additive overhead, explicitly. Chapter 15 gave the FPR-zero counterpart for ranked retrieval, where the error budget is the false-negative side. Part IV exhibited the algebra's bounds as operationally achievable, not merely necessary.

Part V drew the boundary and turned the algebra to use. Chapter 16 catalogued where the framework fails, when the rates are not iid, the channel non-stationary, or the inputs adversarial, and what replaces the broken machinery. Chapter 17 took the algebra to the cryptographic setting, cipher maps generalizing the atoms to the trapdoor-computing paradigm with the chapter 7 PPV re-read as the confidentiality measure. Chapter 18, the present chapter, instantiated cipher maps for a working application, encrypted search with Bloom-filter indexing, where the rate-prevalence triangle prices the trade-off between recovery and leakage.

What comes next

The framework is not exhausted by the eighteen chapters. Three threads point past its current scope: approximate query languages that lift the chapter 12 relational algebra to a declarative interface with framework-priced execution costs; federated computation, where the cipher-map machinery generalizes to multiple parties each holding partial trapdoor keys; and the relation to differential privacy, which uses a structurally similar noise-budget abstraction, with the noise explicitly controlled rather than emergent, and which the framework should ultimately subsume as a constrained sibling. Each is a working extension, not a planned chapter; the book closes with the framework in a state that admits the extensions but does not require them.

Bibliographic Notes

The encrypted-search thesis. The confidentiality analysis and the Bloom-filter construction are developed at length in Towell [Tow14], "On the Confidentiality of Bloom-Filter-Based Encrypted Search," the master's thesis that first laid out the framework's encrypted-search reading in operational detail (key schedule, indexing, query protocol) and developed the adversarial-estimation problem of Section 18.4 as its central concern. The present chap-

ter reorganized the thesis material around the Bernoulli framework the rest of the book builds on; the thesis predates the framework's current formulation and reads as a self-contained application of approximate-membership ideas.

Estimating confidentiality via multi-armed bandits. The MAB-based estimation of Section 18.4 draws on Towell [Tow16], "Estimating encrypted-search confidentiality via multi-armed bandits" (IEEE CloudCom 2016), which frames the estimation problem as a bandit problem and develops the moving-average bootstrap CI as a tighter alternative to the union bound. The chapter summarizes the framing and the worked example, leaving the bootstrap construction and the finite-sample simulation studies to the paper.

The searchable-symmetric-encryption literature. The constructions sit in the searchable-symmetric-encryption (SSE) literature tracing to Song, Wagner, and Perrig [SWP00], "Practical Techniques for Searches on Encrypted Data," the foundational paper that introduced the problem in the form Section 18.1 restated, with a stream-cipher construction whose threat model the present chapter inherits. Curtmola et al. [Cur+06], "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," tightened the security definitions to the adaptive-adversary setting and gave the first sub-linear-time SSE construction; it is the standard reference for SSE security notions and the entry point to the modern subfield. The chapter uses the SSE literature for the threat model and construction style without entering the formal security-definition machinery a cryptographic-engineering treatment requires.

Maximizing confidentiality. The PPV-as-confidentiality identification of Section 18.3 extends a line the cipher-maps manuscript began in Towell [Tow26i] (cited from chapter 17), "Maximizing confidentiality in encrypted search," which developed the confidentiality measure in the encrypted-search setting before the broader trapdoor framing of chapter 17 was in place, treating the trade-off curve quantitatively, giving the closed-form ε^* at a target confidentiality, and addressing the multi-query accumulation that Proposition 18.3.1 states without proof. Where the two diverge the chapter prefers the framework's algebraic reading and leaves the manuscript's information-theoretic supplements for the reader who wants the quantitative refinement.

Chapter 19

The Operator Representation

19.1 What This Chapter Is For

Every preceding chapter developed the framework in one voice. A set is a membership predicate; an approximate set is that predicate made noisy; the errors are the rates ε and ω; the operations are union, intersection, and complement, with rates that compose by the law of total probability. That development is complete on its own terms, and nothing in this chapter revises it.

This chapter reads the same objects in a second voice. Each element of the universe becomes a basis vector, a set becomes a vector built from those, a function becomes a matrix, and a noisy channel becomes a stochastic matrix acting on vectors. The bra-ket and tensor notation of finite-dimensional linear algebra carries the account. The two voices describe one framework, and the point of stating both is that each makes visible what the other leaves implicit.

What the operator view exposes

The set-theoretic development states the composition theorem (Section 3.3) and the Kronecker factorization (Section 4.4) as two separate results, proved by two separate arguments, one about chained gates and one about parallel queries. In the operator view they are one fact read in two geometries: serial composition is a matrix product and parallel querying is a tensor product, and both are products of operators. The composition theorem's $1-\prod_i(1-\eta_i)$

form is then not a combinatorial identity to be rediscovered but the multiplicativity of a product, and the spectral structure that the product form hides becomes available: eigenvalues multiply along a chain, and the error of a parity over independent bits is a product of the channels' eigenvalues. The independence axiom (Axiom 1) acquires a one-line restatement, "the noise operator factors," and parametric parsimony becomes the statement that a factored operator carries the sum of its factors' parameters, not their product. None of this is new content; it is the same content with its linear and spectral structure made explicit.

What the set view keeps

The trade runs the other way as well. The matrix view flattens the combinatorial semantics that the set view carries on its surface. A false-positive rate written as a matrix entry Q_{yx} no longer announces that it is the probability of accepting a non-member; a union written as an elementwise OR on occupation vectors no longer displays the three truth-regions ($A \setminus B$, $B \setminus A$, outside both) that Section 6.2 weights to get the intersection rate right. The operator view can compute these quantities, but it does not keep them legible the way the predicate language does. Where the combinatorial meaning is the point, the set view is the better instrument, and the matrix is the supporting calculation.

One framework, two lenses

So the relationship is parallel and complementary, not hierarchical. A reader who wants the meaning of an operation reaches for the predicate; a reader who wants its algebraic or spectral structure reaches for the operator. The remaining sections build the operator lens in order: the computational basis (Section 19.2), maps as stochastic matrices (Section 19.3), the subtler case of sets as tensored channels (Section 19.4), the algebra and spectra that the view exposes (Section 19.5), and a precise account of how far the formalism coincides with finite-dimensional quantum mechanics and where, at Section 16.5, the coincidence stops (Section 19.6). The bra-ket notation throughout is a notational convenience for the classical stochastic algebra, adopted because it makes the products and factorizations easy to write, and it is not a claim that the framework is quantum mechanical.

19.2 The Computational Basis

Fix a finite universe U with $|U| = N$ elements, ordered once and for all. The operator view begins by giving each element an axis.

Definition 19.2.1 (Computational basis)**.** *To each element $x \in U$ assign the basis vector $|x\rangle \in \mathbb{R}^N$ that is 1 in the coordinate indexed by x and 0 elsewhere. The family $\{|x\rangle : x \in U\}$ is the* computational basis *of $\mathbb{R}^N$: N mutually orthogonal one-hot vectors, one per element.*

The name is borrowed from the measurement basis of a quantum register, and the borrowing is deliberate; Section 19.6 makes the correspondence precise. For now $|x\rangle$ is a one-hot indicator and nothing more.

A set is its occupation vector

A classical set is built by superposing the basis vectors of its members.

Definition 19.2.2 (Occupation vector)**.** *For $A \subseteq U$ the* occupation vector *(equivalently the indicator vector) is*

$$\mathbb{1}_A \;=\; \sum_{x \in A} |x\rangle \;\in\; \{0,1\}^N,$$

the 0/1 vector whose x-coordinate is 1 exactly when $x \in A$.

For $U = \{a, b, c\}$ in that order, the set $\{a, b\}$ is $\mathbb{1}_{\{a,b\}} = |a\rangle + |b\rangle = (1, 1, 0)^{\mathsf{T}}$, the full universe is $(1, 1, 1)^{\mathsf{T}}$, and the empty set is the zero vector. This is the membership predicate of Part II written in coordinates: the x-coordinate of $\mathbb{1}_A$ is the truth value of "$x \in A$."

An occupation vector is not a probability vector

The single distinction that organizes the rest of the chapter is between two kinds of vector that both live in $\mathbb{R}^N$ and are easy to conflate.

The occupation vector $\mathbb{1}_A$ is a vector of *bits*. Its coordinates are independent yes/no answers, and they sum to

$$\sum_x (\mathbb{1}_A)_x \;=\; |A|,$$

the cardinality of A, which is anything from 0 to N. It is a point of the Boolean cube $\{0,1\}^N$, not a distribution.

A *random element* of U is a different object. It is a probability vector

$$p \;=\; \sum_{x\in U} p_x|x\rangle, \qquad p_x \geq 0, \quad \sum_x p_x = 1,$$

a point of the simplex, with p_x the probability that the element is x. A basis vector $|x\rangle$ is the degenerate case, the point mass at x, and is the only vector that is at once a one-hot occupation vector (the singleton $\{x\}$) and a probability vector.

These two readings of $\mathbb{R}^N$ pull the framework in two directions, and the split is exactly the maps-versus-sets split of the next two sections. A *map* sends an input element to an output element, so it transports probability mass on the simplex: its natural operator is one $N \times N$ stochastic matrix, taken up in Section 19.3. A *set*, by contrast, is a vector of bits, and a noisy set flips those bits; flipping a coordinate of the cube is not a motion on the simplex, and modeling it correctly requires a different operator, taken up in Section 19.4. The clean case comes first.

19.3 Maps as Stochastic Matrices

A function is the cleanest thing to put in operator form, because a function moves an element to an element, and an element is a basis vector.

A deterministic function is a $0/1$ matrix

Definition 19.3.1 (Map matrix)**.** *For a function $f : U \to V$ between finite sets, the* map matrix *M_f is the $|V| \times |U|$ matrix with a single 1 in each column, placed so that*

$$M_f|x\rangle \;=\; |f(x)\rangle \qquad \textit{for every } x \in U.$$

Column x of M_f is the one-hot vector $|f(x)\rangle$; entry (y, x) is 1 when $y = f(x)$ and 0 otherwise.

Each column sums to one (a function sends each input somewhere, and to exactly one place), so M_f is *column-stochastic* with entries in $\{0, 1\}$. Applying M_f to a basis vector reads off the function value; applying it to a probability vector p pushes the distribution forward, $M_f p$ being the law of $f(X)$ when $X \sim p$. When $U = V$ and f is a bijection, M_f is a permutation matrix, the operator face of the user's $f(x) = M|x\rangle$ reading of a function: a relabeling of the basis. A non-injective f collapses several basis vectors onto one, and M_f then has a repeated column and is singular.

A Bernoulli map is a column-stochastic matrix

Allowing the columns to be distributions rather than point masses turns a function into an approximate one.

Definition 19.3.2 (Bernoulli map matrix)**.** *A* Bernoulli map $\tilde{f} : U \to V$ *is given by a column-stochastic matrix Q of shape $|V| \times |U|$,*

$$Q_{yx} \;=\; \mathrm{P}\left(\tilde{f}(x) = y\right), \qquad Q_{yx} \geq 0, \quad \sum_{y} Q_{yx} = 1,$$

so that column x, namely $Q|x\rangle$, is the output distribution on V produced by the input x. The model order *(Definition 2.4.1) is the number of distinct columns of Q, the number of input blocks that produce genuinely different output distributions.*

A deterministic f is the special case in which every column is one-hot, so Definition 19.3.1 sits inside Definition 19.3.2. The matrix here is the n-ary channel of Definition 9.2.1, read as an operator. The two definitions are the same object under transposition: the channel of Section 9.2 is written row-stochastic, with input blocks indexing rows and the matrix acting on row vectors from the right, the natural convention when one reads the matrix as a confusion table; the operator here is written column-stochastic, with inputs indexing columns and the matrix acting on column vectors $|x\rangle$ from the left, the natural convention when one reads the matrix as a map. The entries are the same conditional probabilities $\mathrm{P}\,(\text{output} \mid \text{input})$, and "column-stochastic operator" and "row-stochastic channel" name one matrix in two layouts. The binary case $|U| = |V| = 2$ recovers the 2×2 channel of Part II, with $Q|0\rangle$ and $Q|1\rangle$ the two columns carrying $(1-\varepsilon, \varepsilon)$ and $(\omega, 1-\omega)$.

Composition is the matrix product

The reason to write a map as a matrix is that composing maps becomes multiplying matrices.

Proposition 19.3.1 (Serial composition of Bernoulli maps)**.** *Let $\tilde{f} : U \to V$ and $\tilde{g} : V \to W$ be Bernoulli maps with column-stochastic matrices Q_f and Q_g, and suppose the noise of $\tilde{g}$ is independent of the noise of $\tilde{f}$. Then the composite $\tilde{g} \circ \tilde{f} : U \to W$ is a Bernoulli map with column-stochastic matrix*

$$Q_{g \circ f} \;=\; Q_g\, Q_f, \qquad (Q_g Q_f)_{wx} \;=\; \sum_{v \in V} (Q_g)_{wv}\, (Q_f)_{vx}.$$

Proof. The composite output law at input x is $\mathrm{P}\left(\tilde{g}(\tilde{f}(x)) = w\right)$. Conditioning on the intermediate value $v = \tilde{f}(x)$ and using the independence of the two noise sources,

$$\mathrm{P}\left(\tilde{g}(\tilde{f}(x)) = w\right) = \sum_{v \in V} \mathrm{P}\left(\tilde{g}(v) = w\right) \mathrm{P}\left(\tilde{f}(x) = v\right) = \sum_{v \in V} (Q_g)_{wv} (Q_f)_{vx},$$

the (w, x) entry of $Q_g Q_f$. Each column sums to one because Q_f's column x is a distribution and Q_g maps distributions to distributions ($\sum_w \sum_v (Q_g)_{wv}(Q_f)_{vx} = \sum_v (Q_f)_{vx} = 1$), so $Q_g Q_f$ is column-stochastic and is therefore again a Bernoulli map. □

This is the operator face of serial channel composition (Section 9.3), transposed to the column-stochastic layout (the row-stochastic statement there is $Q_f Q_g$, the same product read in the other order). The composition theorem of Theorem 3.3.1 is its specialization to the one-sided error direction. Take two one-sided stages on a shared codomain, each correct with probability $1 - \omega_i$ on a true input and never producing a spurious accept. The relevant entry of the product $Q_g Q_f$ is the probability that both stages pass the truth through, which by the product is $(1 - \omega_1)(1 - \omega_2)$, so the composite miss rate is

$$\omega_{g \circ f} \;=\; 1 - (1 - \omega_1)(1 - \omega_2),$$

and for a k-stage chain $\omega_{\text{total}} = 1 - \prod_i (1 - \omega_i)$, exactly Equation (3.5). The set-theoretic chapter proved this by induction on the chain; the operator view gets it as one diagonal entry of a matrix product, and the inductive step is the associativity of matrix multiplication. The two derivations agree because they are the same derivation in two notations.

Deterministic maps as a sanity check

The product specializes correctly at the deterministic end. If f and g are ordinary functions, then $M_g M_f |x\rangle = M_g |f(x)\rangle = |g(f(x))\rangle = |(g \circ f)(x)\rangle$, so $M_{g \circ f} = M_g M_f$: the product of two permutation matrices is the permutation of the composite, and the product of two 0/1 map matrices is the 0/1 matrix of the composed function. The approximate composition law degenerates to function composition exactly when the columns degenerate to point masses, which is the consistency one wants between the noisy theory and the exact one it generalizes. The next section turns to the case where this clean single-matrix picture is the wrong one.

19.4 Sets as Tensored Channels

The map of Section 19.3 was clean because a map moves an element, and an element is one basis vector, so one stochastic matrix carries the whole story. A set is not like that, and the difference is the most important pedagogy in this chapter. A noisy set does not move an element; it flips membership bits, one per element of the universe, and that is a different operation requiring a different operator.

The temptation, and why it is wrong

The occupation vector $\mathbb{1}_A$ of Definition 19.2.2 lives in $\mathbb{R}^N$, and a Bernoulli map of Definition 19.3.2 is an $N \times N$ stochastic matrix acting on $\mathbb{R}^N$. It is tempting to join them: pick one $N \times N$ stochastic matrix Q and write the noisy set as $Q\,\mathbb{1}_A$. This is wrong, and seeing why fixes the whole picture.

A column-stochastic Q acting on $\mathbb{R}^N$ transports probability mass among the N coordinates: it sends a distribution over *which single element* to another such distribution. Its proper input is a probability vector on the simplex, a random element. But $\mathbb{1}_A$ is not a random element; it is a vector of N independent bits that sums to $|A|$, not to 1. Feeding it to Q asks Q to move membership mass from one element to another, as if a member of A could "leak" into being a member somewhere else. That is not what noise does to a set. Set noise acts on each element *separately*: for each x, the bit "$x \in A$" is reported correctly or flipped, and the flip at x says nothing about the bit at any other element. One $N \times N$ matrix on the occupation vector models the wrong dynamics, the evolution of a random element rather than the independent corruption of N membership bits.

One two-dimensional space per element

The correct operator keeps the elements apart. Each element carries its own two-state membership question, in or out, so it carries its own two-dimensional space.

Definition 19.4.1 (Per-element membership space)**.** *For each $x \in U$ let $\mathbb{C}^2_x$ be a two-dimensional space with basis $\{|0\rangle_x, |1\rangle_x\}$, where $|1\rangle_x$ encodes "x is a member" and $|0\rangle_x$ encodes "x is not." A configuration of the whole universe, a definite membership status for every element, is a basis vector of the tensor product*

$$\mathcal{V} \;=\; \bigotimes_{x \in U} \mathbb{C}^2_x \;\cong\; (\mathbb{C}^2)^{\otimes N}, \qquad \dim \mathcal{V} = 2^N,$$

namely $\bigotimes_{x\in U}|b_x\rangle$ *with* $b_x \in \{0,1\}$ *the membership bit of* x.

The 2^N basis vectors of $\mathcal{V}$ are in bijection with the 2^N subsets of U: the subset A is the configuration $\bigotimes_x|(\mathbb{1}_A)_x\rangle$, member bits set to $|1\rangle$ and the rest to $|0\rangle$. This is the same set, now encoded as a tensor of bits rather than packed into a single occupation vector. The occupation vector $\mathbb{1}_A \in \{0,1\}^N$ is the *compact encoding*, a list of the bits; the tensor basis vector $\bigotimes_x|(\mathbb{1}_A)_x\rangle \in \mathcal{V}$ is the *configuration*, the same bits placed in N separate two-level systems so that an operator can act on each independently. The point of moving to the 2^N-dimensional space is precisely that it gives each membership bit its own factor for the noise to act on.

The noise operator is a Kronecker product

Per-element noise is now a per-factor channel.

Definition 19.4.2 (Set noise operator)**.** *For each element* x *let* Q_x *be the* 2×2 *column-stochastic single-bit channel*

$$Q_x \;=\; \begin{pmatrix} 1-\varepsilon_x & \omega_x \\ \varepsilon_x & 1-\omega_x \end{pmatrix},$$

whose columns are indexed (left to right) by the true status "out" then "in": column "out" reports membership with probability ε_x *(a false positive), column "in" reports non-membership with probability* ω_x *(a false negative). The* set noise operator *is the Kronecker product*

$$Q_{\text{set}} \;=\; \bigotimes_{x\in U} Q_x,$$

a $2^N \times 2^N$ *column-stochastic matrix on* $\mathcal{V}$.

Acting on a configuration, Q_{set} flips each membership bit through its own channel and reports the joint result. Its $(\,\cdot\,, A)$ column is the distribution over reported subsets when the true set is A: by the Kronecker structure that distribution is the product over elements of the per-element report laws, member bits surviving with probability $1-\omega_x$ and non-member bits staying silent with probability $1-\varepsilon_x$. Under the first-order homogeneous model every Q_x equals one channel Q with rates (ε,ω), and $Q_{\text{set}} = Q^{\otimes N}$.

This Kronecker product is Axiom 1

The tensor structure is not a new modeling choice. It is element-wise independence, restated as a property of an operator.

Proposition 19.4.1 (The set operator factors exactly when Axiom 1 holds)**.** *The joint membership-report channel of a Bernoulli set is a Kronecker product of per-element* 2×2 *channels,* $Q_{\text{set}} = \bigotimes_x Q_x$*, if and only if the error events* $\{E_x : x \in U\}$ *are mutually independent, that is, if and only if Axiom 1 holds.*

Proof. If Axiom 1 holds, the per-element reports are mutually independent given the true set, so the joint report law factors across elements; the $(\,\cdot\,, A)$ column of the joint channel is the product of the per-element columns, which is the corresponding column of $\bigotimes_x Q_x$. Conversely, if $Q_{\text{set}} = \bigotimes_x Q_x$, then every column factors as a product of two-dimensional marginals, so the reports are independent across elements, which is Axiom 1. A non-factoring joint channel (one with correlation across elements, as in Section 4.2) is by definition not a Kronecker product of 2×2 matrices. □

This is the Kronecker factorization theorem (Theorem 4.4.1) read as the defining property of the set's noise operator rather than as a statement about m parallel queries. The chapter-4 theorem said the joint confusion matrix of independent queries factors as a tensor product; Proposition 19.4.1 says the noise operator of an entire set *is* that tensor product, and that this factorization is logically the same statement as Axiom 1. Where the set view writes "the errors at distinct elements are independent," the operator view writes "the channel is a product operator," and the two are interchangeable.

Maps and sets live in different spaces

The contrast with Section 19.3 is now exact, and it is the heart of the chapter.

A *map* on U acts on $\mathbb{C}^N$, one space of dimension N, by one stochastic matrix; its inputs are random elements (simplex vectors). A *set* on U acts on $(\mathbb{C}^2)^{\otimes N}$, the tensor of N two-dimensional spaces of total dimension 2^N, by a tensored stochastic matrix; its inputs are configurations of N membership bits. The map moves one element among N possibilities; the set flips N bits, each among 2 possibilities. The occupation vector is a convenient *encoding* of the set's state, but the object that evolves the state is the 2^N-dimensional tensored operator, not an $N \times N$ matrix on the occupation vector. Conflating the two, the temptation this section opened with, is exactly the confusion of a vector of bits with a probability vector that Section 19.2 warned about.

Parsimony is the factorization

The two axioms bought parametric parsimony in chapter 4, and in operator terms that purchase is the factorization itself. An arbitrary column-stochastic $2^N \times 2^N$ matrix on $\mathcal{V}$ has 2^N columns, each a distribution over 2^N reported subsets, hence $2^N(2^N - 1)$ free parameters, a number that is hopeless even for modest N. The factored operator $\bigotimes_x Q_x$ is fixed by its N factors, each a 2×2 channel with two free rates, for $2N$ parameters in all (and just 2 in the homogeneous case). The reduction from $2^N(2^N - 1)$ to $2N$ is the operator reading of Corollary 4.4.1.2: the parameters add across factors instead of multiplying across configurations, because the operator factors. Parsimony and factorization are one fact, and Axiom 1 is what supplies it.

19.5 Composition, Spectra, and the Algebra

With maps as matrices and sets as tensored channels in place, three things the set-theoretic chapters proved separately become facets of one algebraic picture. This section collects them, in each case setting the operator statement beside the predicate statement it matches.

Independence is a product, and parsimony is the factoring

The first facet is a recap in one sentence. Proposition 19.4.1 identified element-wise independence with a product operator: the set view's "the errors at distinct elements are independent" is the operator view's "$Q_{\text{set}} = \bigotimes_x Q_x$." The parsimony that followed, $2N$ parameters in place of $2^N(2^N - 1)$, is the arithmetic of a factored operator, whose parameter budget is the sum of its factors' budgets. Independence, factorization, and parsimony are three readings of the single equation $Q_{\text{set}} = \bigotimes_x Q_x$.

Composition multiplies eigenvalues

The second facet sharpens Proposition 19.3.1. Serial composition of Bernoulli maps is the matrix product $Q_g Q_f$, and a matrix product does more than compose: when the factors are simultaneously diagonalizable, it multiplies eigenvalues. Each one-sided survival probability $1 - \eta_i$ is an eigenvalue of stage i on the eigenvector that carries the truth through unflipped, and the chain's survival probability $\prod_i (1 - \eta_i)$ is the product of those eigenvalues.

The composition theorem's

$$\eta_{\text{total}} \;=\; 1 - \prod_{i=1}^{k} (1 - \eta_i) \qquad \text{(Equation (3.5))}$$

is therefore a spectral statement: one minus a product of eigenvalues. The set view obtained it by induction over chained gates; the operator view reads it off the spectrum of a product. The next facet makes the spectral reading do work the combinatorial form does not display.

Parity: one quantity under three names

The cleanest place the spectrum earns its keep is the parity of noisy bits. Take the symmetric single-bit flip channel

$$Q(q) \;=\; \begin{pmatrix} 1-q & q \\ q & 1-q \end{pmatrix},$$

which flips a bit with probability q.

Definition 19.5.1 (The nontrivial eigenvalue of a flip channel)**.** *The symmetric flip channel $Q(q)$ has eigenvectors $(1,1)^{\mathsf{T}}$ and $(1,-1)^{\mathsf{T}}$ with eigenvalues 1 and $1-2q$. The first eigenvalue, on the all-ones vector, is the stochasticity of the channel and carries no information; the second, $\lambda(q) = 1-2q$, is the* nontrivial eigenvalue*, the factor by which the channel contracts the sign coordinate $(1,-1)^{\mathsf{T}}$.*

The number $\lambda(q) = 1 - 2q$ has two further identities that coincide with the eigenvalue. Read the bit through the sign map $b \mapsto (-1)^b$, sending $\{0,1\}$ to $\{+1,-1\}$. The expected sign after one flip is

$$\mathrm{E}\left[(-1)^{\text{flip}}\right] = (1-q)(+1) + q(-1) = 1 - 2q,$$

the ± 1 *bias* of the channel. And $1-2q$ is the order-one *Walsh (Fourier) coefficient* of the channel on the Boolean cube, the coefficient of the single-bit character $\chi(b) = (-1)^b$ under the expansion. Eigenvalue, ± 1 bias, and Walsh coefficient are three names for $1-2q$, and the names matter because each makes the parity formula obvious from a different direction.

Proposition 19.5.1 (Spectral error rate of a noisy parity)**.** *Let $b_1, \ldots, b_n$ be bits, each flipped independently by a symmetric channel of rate q_i, and*

let the output be the parity (XOR) $b_1 \oplus \cdots \oplus b_n$. The parity is wrong with probability

$$\mathrm{P}\,(\textit{parity flipped}) \;=\; \frac{1}{2}\Big(1 - \prod_{i=1}^{n}(1-2q_i)\Big) \;=\; \frac{1}{2}\Big(1 - \prod_{i=1}^{n}\lambda(q_i)\Big),$$

half of one minus the product of the channels' nontrivial eigenvalues.

Proof. The parity flips iff an odd number of the n bits flip. Pass to signs: the output sign is $\prod_i (-1)^{b'_i}$ where b'_i is the flipped bit, and the parity is wrong exactly when the product of flip signs is -1. By independence the expected product of flip signs factors,

$$\mathrm{E}\left[\textstyle\prod_i (-1)^{\mathrm{flip}_i}\right] = \displaystyle\prod_{i=1}^{n} \mathrm{E}\left[(-1)^{\mathrm{flip}_i}\right] = \prod_{i=1}^{n}(1-2q_i),$$

each factor the ± 1 bias of one channel. Writing P for the probability the parity is flipped, the expected sign of the error is $(+1)(1-P) + (-1)P = 1 - 2P$, so $1 - 2P = \prod_i (1-2q_i)$ and $P = \frac{1}{2}(1 - \prod_i (1-2q_i))$. Each $1-2q_i$ is the nontrivial eigenvalue $\lambda(q_i)$ of Definition 19.5.1, giving the spectral form. □

The product of ± 1 biases is the multiplicativity of the order-one Walsh coefficient under XOR, and it is the same multiplicativity as eigenvalues under a tensor product, since the parity error operator is built from the per-bit channels by the Kronecker product of Definition 19.4.2 restricted to the sign sector. This identity is the syndrome-bit calculation used in the quantum-readout setting, where a measured parity (a stabilizer syndrome) inherits $\prod_i (1-2q_i)$ from its constituent single-qubit readout errors, and it is the classical noisy-formula tradition begun for redundant computation by von Neumann and analyzed for bounded-depth circuits by Pippenger, where the ± 1-bias product is the standard accounting for error through XOR gates. The operator view unifies them: eigenvalue, Walsh coefficient, and ± 1 bias are one quantity, and a parity of independent noisy bits contracts the sign sector by the product of that quantity over the bits.

Set operations in the tensor algebra

The last facet returns to the set algebra of Chapter 6 and reads it on configurations. Union and intersection of approximate sets are bitwise logic on membership configurations: the union $\mathsf{A}^{\pm} \cup \mathsf{B}^{\pm}$ (Definition 6.1.1) is the OR

of the two membership bits at each element, and the intersection $\mathsf{A}^{\pm} \cap \mathsf{B}^{\pm}$ (Definition 6.2.1) is the AND. On the tensor space $\mathcal{V}$ both are elementwise (per-factor) Boolean operations on the reported configurations, the predicate-level operations of chapter 6 written one factor at a time.

The operator view does not, however, erase the subtlety chapter 6 was careful about. The intersection's false-positive rate is not the bare product $\varepsilon_A \varepsilon_B$; it is the partition-weighted three-region sum

$$\varepsilon_{\cap} = w_1(1-\omega_A)\varepsilon_B + w_2\varepsilon_A(1-\omega_B) + w_3\varepsilon_A\varepsilon_B \qquad \text{(Equation (6.5))}$$

of Proposition 6.2.1, where w_1, w_2, w_3 weight the regions $A \setminus B$, $B \setminus A$, and outside both. The product $\varepsilon_A\varepsilon_B$ is only the third term, the both-outside corner, exact in the sparse limit where almost every non-member of $A \cap B$ lies outside both operands. The reason survives translation: a false positive of the intersection requires both operands to report membership at an element where the intersection is absent, and that element's true status differs across the three regions, so the per-element AND consults different entries of Q_A and Q_B depending on the region. The elementwise tensor algebra computes the rate correctly only when it is fed the right per-element truth, which is the same bookkeeping the predicate derivation does by hand. The union is the De Morgan dual, its false-negative rate the three-region sum Equation (6.2) and its false-positive rate the clean product $1-(1-\varepsilon_A)(1-\varepsilon_B)$ of Equation (6.1). The operator picture makes the logic per-factor and the composition multiplicative; it leaves the region accounting exactly where chapter 6 put it.

19.6 The Classical Sector of a Quantum Formalism

The notation of this chapter, kets, tensor products, eigenvalues, is the notation of finite-dimensional quantum mechanics, and the resemblance is close enough that the relationship deserves a precise statement. The framework is the classical sector of the quantum formalism: the same linear algebra, on probabilities rather than amplitudes, restricted to the diagonal of the state. This section says exactly what coincides, what does not, and why the bra-ket notation is a convenience and not a claim.

Same algebra, different objects

Set the two formalisms side by side. Finite-dimensional quantum mechanics works in $\mathbb{C}^d$, builds composite systems by the tensor product $\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$,

evolves states by linear operators, and reads spectra off those operators. The operator view of this chapter does all four: it works in $\mathbb{C}^N$ for maps and $(\mathbb{C}^2)^{\otimes N}$ for sets (Section 19.2, Definition 19.4.2), composes systems by the Kronecker product, evolves by stochastic matrices, and reads error rates off eigenvalues (Proposition 19.5.1). The linear-algebraic machinery is identical.

The objects acted on are not. A quantum state vector holds probability *amplitudes*, complex numbers whose squared magnitudes are probabilities, and amplitudes can be negative or complex and so can cancel. The vectors here hold probabilities directly (a simplex vector for a random element) or bits (a 0/1 occupation vector for a set), and probabilities only accumulate. Two consequences of quantum mechanics are therefore simply absent. There is no Born-rule square: a coordinate of a vector here *is* a probability, not an amplitude to be squared into one. And there is no measurement collapse: reading a coordinate does not disturb the others, because the vector is already a classical description, not a superposition awaiting measurement. The chapter uses kets to write one-hot indicators and tensor products to write independence, and that is the whole of the borrowing.

The framework lives on the diagonal

The precise locating statement uses the density operator. A quantum system in $\mathbb{C}^d$ is described not only by a state vector but, more generally, by a density matrix ρ, a positive operator of unit trace whose diagonal entries ρ_{ii} are the probabilities of the basis outcomes and whose off-diagonal entries ρ_{ij} are the coherences that encode interference. The Bernoulli framework is exactly the classical stochastic algebra of the *diagonal* of ρ in the computational basis. A probability vector here is the diagonal of a density matrix; a stochastic matrix here is the map induced on those diagonals by a quantum channel that does not mix the diagonal with the off-diagonal part. This is the readout sector of the quantum-readout instance in Section 9.2: measuring n qubits in the computational basis is an n-ary channel (Definition 9.2.1) whose assignment matrix is precisely the stochastic operator on diagonals, and when per-qubit readout errors are independent it is the Kronecker product of single-qubit channels, the same factorization Definition 19.4.2 gives a set. The framework is the faithful description of that diagonal, and only of it.

Where the diagonal description stops being faithful

A diagonal description is not always available, and the boundary is the subject of Section 16.5. When the state carries coherence, when ρ has nonzero off-diagonal entries, the diagonal alone does not determine the measurement statistics across incompatible bases, and there is no consistent assignment of definite values to all observables to which a stochastic matrix could be applied. That is the regime where no latent value exists, and Section 16.5 shows, through Fine's theorem and the Bell and Kochen-Specker obstructions, that it is a falsifiable physical fact and not a modeling convenience. In operator terms the framework's vectors are diagonals, the diagonal is all the framework can see, and contextuality is exactly the content of the off-diagonal coherences the diagonal omits. Decoherence, the physical process that zeroes the off-diagonals, is what hands the framework a genuine diagonal to model; on a decohered or classically prepared state the operator view of this chapter is exact, and off it the view is blind to the coherences.

The other half of the boundary is where the square returns. Section 11.3 compared classical amplification, which composes probabilities and reaches near-certainty in $O(1/p)$ repeated looks, with amplitude amplification, which composes amplitudes and reaches it in $O(1/\sqrt{p})$ steps. The quadratic gap is the Born-rule square applied to a quantity that accumulates linearly, and it appears precisely because amplitude amplification works on the off-diagonal, amplitude-bearing part of the state that the diagonal description discards. The framework's amplification theorems are the diagonal-sector statements; the quantum speedup is what the off-diagonal sector buys, and it is unavailable to a formalism that keeps only the diagonal.

The notation is a convenience

The honest summary is therefore narrow. Adopting bra-ket and tensor notation in this chapter is a notational convenience for the classical stochastic algebra: it makes serial composition a product (Proposition 19.3.1), independence a factorization (Proposition 19.4.1), and a parity error rate a product of eigenvalues (Proposition 19.5.1), all of which are awkward to write in the predicate language and natural in the operator one. It is not a claim that the framework is quantum mechanical. The framework is the classical, diagonal, real-and-nonnegative shadow of the quantum formalism; Section 19.1 opened by calling the operator view a second lens on the same set-theoretic content, and the precise version of that claim is this: the lens is finite-dimensional linear algebra, the same linear algebra quantum mechan-

ics uses, pointed at the diagonal where probabilities live and amplitudes do not.

Bibliographic Notes

Kronecker products and finite-dimensional linear algebra. The Kronecker product that carries the tensored-channel construction of Proposition 19.4.1, together with its mixed-product rule and its action on spectra, is treated in Horn and Johnson [HJ91], *Topics in Matrix Analysis*, the standard reference for the operator identities this chapter uses. The eigenvalues-multiply-along-a-product reading of serial composition is the elementary spectral theory of commuting operators in the same source.

Stochastic matrices and Markov chains. A Bernoulli map is a column-stochastic matrix, and the algebra of such matrices, products, stationary vectors, and spectra, is the algebra of finite Markov chains. Levin, Peres, and Wilmer [LPW17], *Markov Chains and Mixing Times*, is the modern textbook treatment; Mitzenmacher and Upfal [MU17], *Probability and Computing*, gives the same material in the algorithmic setting closest to this book's use of it. The transpose convention, whether the stochastic matrix acts on the left of column vectors or the right of row vectors, differs across these sources and from Section 9.2; the underlying matrix is the same and only its layout changes.

Fourier analysis on the Boolean cube. The ± 1-bias and Walsh-coefficient readings of the flip eigenvalue in Proposition 19.5.1 belong to the Fourier-analysis-of-Boolean-functions tradition, for which O'Donnell [ODo14], *Analysis of Boolean Functions*, is the comprehensive reference. The multiplicativity of the order-one coefficient under XOR, the identity behind the noisy-parity rate, is the cube-Fourier face of the tensor factorization of Proposition 19.4.1.

Reliable computation from noisy components. The accounting of error through XOR gates by the product of ± 1 biases is the classical tradition of computing reliably with unreliable parts. Neumann [Neu56], "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," began it, and Pippenger [Pip88], "Reliable Computation by Formulas in the Presence of Noise," gave the formula-depth analysis in which the bias-product bookkeeping is standard. The syndrome-bit version of the

same identity, a measured parity inheriting $\prod_i (1 - 2q_i)$ from its constituent single-bit readout errors, is the quantum-readout instance of Section 9.2, developed in the companion paper [Tow26a].

Density matrices and the diagonal sector. The locating statement of Section 19.6, that the framework is the classical stochastic algebra on the diagonal of the density operator in the computational basis, rests on standard density-matrix and decoherence facts. Nielsen and Chuang [NC10], *Quantum Computation and Quantum Information*, is the textbook reference for the density operator, the distinction between its diagonal probabilities and its off-diagonal coherences, and the readout-error channel that this chapter reads as a Kronecker product of single-qubit stochastic matrices.

Appendix A

Probability Refresher

Placeholder. Content drafted in Plan 5.

Appendix B

The Five Core Principles

Placeholder. Cheat-sheet of the Two Axioms, Parametric Parsimony, Space-Accuracy Duality, Composition Closure, and the Boolean Thread. Content drafted in Plan 5.

Appendix C

Library Walkthrough

Placeholder. Tour of the `bernoulli` *Python package as a unified entry point. Content drafted in Plan 5.*